Estimating Nonlinear Heterogeneous Agents Models with Neural Networks[*]

Hanno Kase
University of Minnesota

Leonardo Melosi
FRB Chicago
CEPR

Matthias Rottner
Deutsche Bundesbank

February 27, 2023

Abstract

Economists typically make simplifying assumptions to make the solution and estimation of their highly complex models feasible. These simplifications include approximating the true nonlinear dynamics of the model, disregarding aggregate uncertainty or assuming that all agents are identical. While relaxing these assumptions is well-known to give rise to complicated curse-of-dimensionality problems, it is often unclear how seriously these simplifications distort the dynamics and predictions of the model. We leverage the recent advancements in machine learning to develop a solution and estimation method based on neural networks that does not require these strong assumptions. We apply our method to a nonlinear Heterogeneous Agents New Keynesian (HANK) model with a zero lower bound (ZLB) constraint for the nominal interest rate to show that the method is much more efficient than existing global solution methods and that the estimation converges to the true parameter values. Further, this application sheds light on how effectively our method is capable to simultaneously deal with a large number of state variables and parameters, nonlinear dynamics, heterogeneity as well as aggregate uncertainty.

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: kase0060@umn.edu, leonardo.melosi@chi.frb.org, matthias.rottner@bundesbank.de. We thank Jesús Fernández-Villaverde, David Levine, Galo Nuño, and Frank Schorfheide for their very helpful suggestions. We thank our discussants and seminar participants at the Banque de France, Deutsche Bundesbank, the Schumpeter-BSE-Seminar in Berlin, CEBRA 2022 Annual Meeting, Conference on advanced analytics by the ECB, BoE and DAFM at King's College London, the European University Institute, Conference on Non-traditional Data, Machine Learning and Natural Language Processing in Macroeconomics at Sveriges Riksbank, the Ghent University Workshop on Empirical Macroeconomics, the 29th Symposium of the SNDE, SEM Annual Conference 2022, EEA Conference 2022, Midwest Macro Meeting 2022, and 4th EUI Alumni Conference in Economics. 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 Federal Reserve Bank of Chicago or any other person associated with the Federal Reserve System.

1 Introduction

Modern economic models are often hard to solve, forcing economists to study tractable approximations of these models and limiting their empirical analysis. However, this tractability often comes at the cost of losing interesting features of the model, such as important nonlinearities like the zero lower bound (ZLB) for the nominal interest rate, agents heterogeneity, and stochastic volatility. We propose an approach based on machine learning to solve and estimate models without resorting to approximations or simplifications that necessarily distort their properties and predictions. We then apply our method to estimate a quantitative Heterogeneous Agent New Keynesian (HANK) model in its nonlinear specification.

Our method exploits the many advantages of neural networks. Since neural networks can handle a large number of inputs, they are well-suited to learn the solution mapping of models with many state variables and nonlinear dynamics with great accuracy. In particular, an important property of neural networks is scalability, implying that additional extra inputs (e.g., state variables) can be added at low computational costs. This enables neural networks to obtain a global approximation of macroeconomic models that feature hundreds or thousands of state variables. As a consequence, neural networks can be used to tackle the well-known curse-of-dimensionality phenomenon that conventional global solution methods face. Even though increasing the complexity of the problem comes with lower costs, it is still very time-consuming to solve only once such elaborated models with neural networks. This is a particularly challenging issue because an estimation framework usually requires a model to be solved at hundreds of thousands of different parameter combinations. This problem renders a conventional estimation strategy impossible.

We overcome this crucial problem by exploiting the fact that neural networks minimize the issue of curse-of-dimensionality that inevitably arises when handling problems with an extremely large number of inputs. But this would not be enough to achieve our goal of estimating high-dimensional economic models. A critical step we propose in this paper is to use the model's parameter vector as direct input in the neural network. In other words, we treat the parameters as pseudo state variables by exploiting the scalability of neural networks. We then train this extended neural network with the parameters as inputs over the entire parameter space. This step turns out to be critical because it yields the model's solution mapping (i.e., the mapping from the parameter space to the model's equilibrium law of motion) instead of just one point in this solution mapping – associated with the model solution for a single point in the parameter space. Even though treating the model's parameter vector as pseudo state variables makes it more challenging to train this neural network, the computational gains are enormous compared to repeatedly evaluating the model's solution mapping at as many points as needed to get an appreciable evaluation of the likelihood function. The reason behind this result is twofold: the scalability of neural networks and the extraordinary efficiency of modern machine learning software and hardware.

Another computational hurdle is that the likelihood function of nonlinear models needs to be evaluated using a Monte Carlo filter (e.g., the particle filter), which are computationally more costly than linear filters (e.g., the Kalman filter). This issue limits the amount of parameter combinations we can evaluate, restricting considerably the scope of estimation. To overcome this problem, we train an additional neural network that provides a direct mapping from the model parameters to the value of the likelihood function – via the particle filter. This is the so-called *surrogate model*, which is trained only on some selected data points to approximate the outcome of the particle filter in an efficient manner. For this strategy, we calculate the values of the likelihood with the particle filter for different parameter combinations over the parameter space and then use these as data points to train the neural network. Importantly, the training requires only thousands of data points so that this strategy reduces the computation time significantly compared to conventional estimation with typically requires millions of draws. We dubbed this approach as *neural network particle filter*. Once this neural network is trained, it takes us a few milliseconds to evaluate the likelihood of models with hundreds of state variables at one parameter value.

To validate our approach, we provide two proofs of concept and then move to the estimation of the HANK model. First, we demonstrate that our extended neural network with the parameters as pseudo state variables captures the dynamics of a standard linearized NK model, for which the true solution can be derived analytically. Second, we focus on a more complex model with a meaningful nonlinearity and compare the results of our estimation framework to a conventional estimation for nonlinear models. The conventional approach requires to resolve the model at each parameter draw with classical global solution methods and then to employ the particle filter. As the conventional approach restricts severely the potential scope of the estimation, we use a tractable RANK model that features an aggregate nonlinearity (zero lower bound). We establish that estimation results are very similar for both approaches.

Finally, we use our approach to solve and estimate a nonlinear HANK model that contains simultaneously idiosyncratic and aggregate risk. Specifically, the model features idiosyncratic and aggregate shocks as well as individual and aggregate nonlinearities in the form of individual borrowing limits for the agents and a ZLB for the monetary authority, respectively. We include 12 parameters in our estimation. As our method does not restrict us to a specific set of parameters, the estimation also contains parameters that affect directly the heterogeneity of the model. Taken all together, the model features hundreds of state variables and pseudo state variables because we have aggregate shocks, aggregate states, an idiosyncratic shock for each agent and asset holdings for each single agent. To provide a controlled environment for the estimation of the HANK model, we generate simulated aggregate data and assess if the estimation can recover the true parameters. For this, we employ our neural network approach to run a Bayesian estimation with 1 million draws. Importantly, the described neural network based estimation of a nonlinear HANK model is completed in less than two days using a modern desktop computer, which underlines the potential of the method.[1]

We demonstrate that our method recovers the true data generating process of all 12

[1]We use an AMD Ryzen 5 5600X 6-Core processor (CPU) and Nvidia GeForce RTX A4500 (GPU) for the neural network based Bayesian estimation.

parameters using standard aggregate data. However, the identification of the parameter varies strongly over the parameters. Our results suggest that standard aggregate data, such as output growth, inflation or interest rates only marginally help pin down the degree of heterogeneity as the likelihood is rather flat. Our method also allows to analyze interactions between nonlinearities, aggregate uncertainty and heterogeneity.

Literature The paper is connected to the literature that develops global solution methods using neural networks to solve complex dynamic economic models. In particular, we build on the approach introduced by Maliar et al. (2021) to allow estimation of models with hundreds or thousands of state variables. Other exciting approaches that rely on neural networks to solve economic models are developed by Fernández-Villaverde et al. (2020), Valaitis and Villa (2021) and Azinovic et al. (2022), among others. However, none of these papers estimate nonlinear, heterogeneous agents models with likelihood methods.

Likelihood estimation adds an additional important layer of complication in that it requires solving the model at hundred 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. (2019) 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 12 parameters using 3 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 called *neural network particle filter*. Furthermore, unlike that paper, we estimate a HANK model. To our knowledge, we are the first ones 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), and Fernández-Villaverde et al. (2021) rely on neural networks, while Schaab (2020) studies 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 large to make likelihood estimation of these models feasible. To achieve this result, we take advantage of the scalability of neural networks and of 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 a first order or on the assumption of perfect foresight regarding aggregate shocks (Bayer et al., 2020; Lee, 2020; Auclert et al., 2021). A strength of our approach is that it does not impose any restrictions on the set of parameters that can be estimated because our method allows us to solve for the model equilibrium in one step.[2] As a consequence, we can additionally estimate the parameters that affect the (stochastic) steady state of the economy including parameters affecting the heterogeneous agents' problem (e.g., agents' idiosyncratic volatility or borrowing limit). Furthermore, our approach can fully capture interesting interactions between the ZLB, stochastic volatility, aggregate risk, the stationary distribution, and other aggregate nonlinearities.

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. Section 5 concludes the paper.

2 An Estimation Framework Based on Neural Networks

This paper develops an estimation framework that utilizes neural networks to estimate macroeconomic models in its fully nonlinear specification. Neural networks, which are at the core of our method, are the fundamental building block of deep learning, which belongs to the family of machine learning methods.[3] Neural networks are well suited to solve complex nonlinear macroeconomic models because they can handle many inputs and are an efficient method to learn fast about highly complicated mathematical functions or mappings. However, a decisive challenge for the estimation of complex models is that it requires the repeated solving of the model for different parameter combinations. Even though neural networks can solve elaborated models such as nonlinear HANK model, it is computationally infeasible to solve these models sufficiently often with neural networks or other global solution methods.

We overcome this decisive problem for the estimation by exploiting the circumstance that neural networks can handle many inputs without encountering the curse-of-dimensionality. In particular, we include the parameters of the model as pseudo state variables in the neural network. We then solve only once for this extended neural network that includes the parameters as pseudo state variables. The extended neural network allows to directly evaluate the solution of the model for different parameter combinations. Even though the inclusion of parameters as pseudo state variables increases the computation time, it is incomparable to repeatedly solving the model without pseudo state variables.

[2]The estimation methods proposed by those papers solve for the stationary distribution in a first step. In a second step, a perturbation techniques is applied to approximate the aggregate dynamics around the stationary distribution. The estimation itself centers then on this second step, which precludes the estimation of any parameter that affects the stationary distribution.

[3]Appendix A provides a brief summary of neural networks and their key properties for our approach. 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.

An additional challenge for the estimation is related to the computations of the likelihood, which evaluates the fit of the model with the data. In particular, the nonlinearity of the model solution precludes the usage of the Kalman filter. Instead, we employ a particle filter to obtain the likelihood. However, the execution of the particle filter is computationally much more costly than the Kalman filter. This limits the amount of parameter combinations that we can consider and restricts severely the scope of the estimation.

We overcome this limitation by relying on a surrogate model, which is trained on some data points and then provides the outcome of interest in a computationally cheap way.[4] Specifically, we train a new additional neural network as surrogate model to approximate the results of the particle filter. For this strategy, we first calculate the likelihoods for different parameter combinations from the solved parameter space with a particle filter. We then use these calculated likelihoods as data points to train an additional neural network. The training of the neural network requires only thousands of data points, which reduces the computation significantly compared to a conventional estimation with typically millions of draws. We denote this approach as neural network based particle filter. Once we are equipped with the neural network based particle filter, it takes us less than 1ms to evaluate a single draw.

It is instructive to compare our estimation procedure to a conventional approach. In a conventional estimation, the model is first solved with numerical methods for a specific parameter combination. In a second step, a filter evaluates the likelihood of the solved model. Based on the likelihood value, a new parameter combination is considered. A conventional estimation requires the repetition of the solving and filtering steps hundreds of thousand times. However, both steps are computationally costly, which renders this approach infeasible for large and complex models. In contrast to this, our developed approach first solves the model over the entire parameter space and then provides a mapping from the parameters to the likelihood in a quick and efficient manner. Once these neural networks are obtained, it is straightforward and very very fast to execute the estimation. Therefore, the developed neural networks approach creates this new possibility of estimating complex nonlinear models.

2.1 Extended Nonlinear Model Representation

We are interested in solving and estimating dynamic stochastic general equilibrium models in its nonlinear specification. The economy in each period is described by a finite vector of state variables $\mathbb{S}_t$. 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 are the structural parameters of the model Θ. These objects describe the dynamics of the model and can be expressed as transition equation:

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

[4]In physics and engineering, the use of surrogate models is well-established. An application to estimated finance models is Chen et al. (2021).

where f is a nonlinear function. This function f is generally unknown and needs to be solved with numerical methods.

For expressing the solution to this type of models, it is useful to distinguish between state variables and control variables. Control variables ψ_t characterize the optimal policy choices by the agents. The model features O control variables. The decisive step in solving these model is to determine the mapping from the state variables to the control variables, which is given from the policy functions $\psi(\cdot)$:

$$\psi_t = \psi(\mathbb{S}_t|\Theta), \tag{2}$$

where the policy functions are nonlinear and depend on the state variables. Once the control variables are determined, the dynamics of the state variables can be analytically calculated.[5] For this reason, the focus in our description is on the policy functions $\psi(\mathbb{S}_t|\bar{\Theta})$.

The policy functions $\psi(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta})$ are derived as the solution to a function space F:

$$F(\psi(\mathbb{S}_t|\Theta)) = 0, \tag{3}$$

where the function space is derived in line with Euler equation methods.

Key Trick: Incorporating Parameters as Pseudo State Variables The problem in estimating is that we need to solve for the policy function again and again for different parameter values. However, this is very costly and can render estimation of challenging models infeasible. The crucial key is that we want to solve the policy function only once, but at the same time, to account for the entire set of parameters that we want to estimate. The set of parameters can be divided in two subsets:

$$\Theta = \{\tilde{\Theta}, \bar{\Theta}\}, \tag{4}$$

where $\tilde{\Theta}$ is the set of parameters to be estimated and $\bar{\Theta}$ is the set of parameters to be calibrated. We treat the parameters to be estimated as pseudo state variables of the economy. The extended policy function with the pseudo state variables can be written as:

$$\psi_t = \psi\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right), \tag{5}$$

where now the values for the parameters $\tilde{\Theta}$ are treated as pseudo state variables.[6] The next subsection explains how we approximate the extended policy function using neural networks in a deep learning approach.

Incorporating Heterogeneity The approach can also capture models that feature heterogeneous agents (e.g. on the household or firm side) as well as multiple countries, counties,

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

[6]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).

sectors or banks. Heterogeneity often assumes the existence of a continuum of agents, which implies that the distribution of individual states and shocks is infinite:

$$\int \mathbb{S}_t^i d\Omega \quad \text{and} \quad \int \nu_t^i d\Omega, \tag{6}$$

where the superscript i stands for an individual agents.

To map such a scenario in our framework with finite states, the key assumption is that we approximate the continuum of households with a large but finite number of agents L as in Maliar et al. (2021).[7] This allows to capture the continuous distribution with a large but finite number of states.[8] This approach is very appealing for a neural network based procedure as neural networks can overcome the curse-of-dimensionality. The distribution can be summarized as:

$$\left\{\mathbb{S}_t^i\right\}_{i=1}^L \quad \text{and} \quad \left\{\nu_t^i\right\}_{i=1}^L. \tag{7}$$

The state variables and shock can be written as:

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} \quad \text{and} \quad \nu_t = \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}, \tag{8}$$

where the superscript A is concerned for aggregate state variables and shocks. In case of heterogeneity with a finite number of agents, e.g. countries, counties, sectors or banks, this directly defines the state variables without an approximation.

In a similar fashion, we adjust the policy functions:

$$\psi_t^i = \psi^I(\mathbb{S}_t^i, \mathbb{S}_t|\bar{\Theta}) \quad \text{and} \quad \psi_t^A = \psi^A(\mathbb{S}_t|\bar{\Theta}). \tag{9}$$

where we assumed that agents only differ in their state variables and structural shocks so that the same policy function ψ^I can be used for all agents if conditioned on the individual variables additionally.

We can now rewrite the transition equation and policy functions as:

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} = f\left(\left\{\left\{\mathbb{S}_{t-1}^i\right\}_{i=1}^L, \mathbb{S}_{t-1}^A\right\}, \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}, \tilde{\Theta}|\bar{\Theta}\right) \tag{10}$$

$$\psi_t = \left\{\left\{\psi_t^i\right\}_{i=1}^L, \psi_t^A\right\} = \left\{\left\{\psi^I(\mathbb{S}_t^i, \mathbb{S}_t|\bar{\Theta})\right\}_{i=1}^L, \psi^A(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta})\right\} \tag{11}$$

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

[7] Agents assume in their maximization problem that their individual weight is zero.

[8] The approach is related to Le Grand and Ragot (2021), where heterogeneity is captured in form of a truncated-history of idiosyncratic shocks. Their method requires that the past realizations depend on an arbitrary but finite number of states, which requires that the idiosyncratic shock needs to be discretized. Our approach refrains from discretizing, and as a consequence, there are no agents with exactly the same history.

2.2 Extended Neural Network-Based Solution Method

We use neural networks to solve the extended policy functions. In particular, we incorporate the parameters to be estimated as pseudo state variables as new elements in a neural network solution algorithm based on Maliar et al. (2021). The difference is remarkable. The obtained extended neural network from our approach provides the numerical solution for all macroeconomic model that are covered by the parameter space. Normally, a procedure without the pseudo state variables gives just the solution for one single combination of parameters. In other words, our approach solves an infinite amount of models instead of one single model.

While the additional state variables would create a problem for classical global solution method due to the curse-of-dimensionality, our approach relies on the scalability of the neural network in handling many inputs. Therefore, the additional burden of including the parameters as pseudo state variables is by far not comparable to resolving the model for each parameterization. We provide a brief summary of neural networks and their key properties that are so handy for our approach in Appendix A.

Extended Neural Networks and Policy Functions We use neural networks to approximate the individual and aggregate policy functions, ψ^I and ψ^A.[9] In particular, we set up two neural networks, ψ^I_{NN} and ψ^A_{NN}, that approximate the individual and aggregate policy, respectively. We train the neural networks to map the inputs, which consists of S state variables and P parameters to be estimated, into a number of output variables O (policy functions). This provides then the numerical solution for all macroeconomic model that are covered by the parameter space. The mapping from the state variables and parameters to the control variables with neural networks as function approximator is:[10]

$$\psi^i_t = \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right), \tag{12}$$

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

The entire vector of control variables is given as:

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

For simplicity, we define $\psi_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) \equiv \left\{\left\{\psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}^L_{i=1}, \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}$. This formulation highlights that our approach solves simultaneously for the individual law of motion and the aggregate law of motion.[11]

[9]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.

[10]The number of neural networks can be adjusted according to the needs of the model. In the outlined case with individual and aggregate policy functions, it is very handy to choose two neural networks.

[11]This differs to the aggregation approach of Krusell and Smith (1998), as discussed in detail in Maliar and Maliar (2020).

Loss Function and Training of Neural Networks The next step is to train the extended neural network to approximate the policy functions. The neural networks ψ^I_{NN} and ψ^A_{NN} are trained to minimize a defined loss function. In particular, we minimize the residual error of a set of equations, where the equations are chosen in line with the Euler equation method. It should be noted that in a setup with heterogeneity, the entire set of equilibrium conditions contains individual and aggregate intertemporal optimality conditions, transition equations, equilibrium conditions, etc. The number of equations that are minimized in line with the Euler equation method corresponds to the number of policy functions $O = O^i \times L + O^A$. The loss function is the sum of the square of the equations from the Euler equation method.[12]

Furthermore, the neural networks ψ_{NN} are trained on a batch with size B. This can be thought of having B economies operating in parallel that are used to train the neural network. As a consequence, we minimize $B \times (O^i \times L + O^A)$ equations in each iteration using deep learning techniques.[13] 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. 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 our solution algorithm that approximates the ergodic distribution of the model. After training the neural network with the given draw of state variables over the batch B in the current iteration, we simulate each economy (batch) for T^{sim} periods forward. The end point of the simulation gives than 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 additionally relies on the antithetic variate to increase the precision.[14] The approach handles hundreds of shocks and is well suited to evaluate expectations in a stochastic setup with randomly drawn shocks.

Parameter Space We train the neural network for the entire parameter space that we consider. We restrict each parameter that is estimated to lie inside some bounds:

$$\tilde{\Theta} = \left\{ \left[\tilde{\Theta}^{\underline{1}}, \tilde{\Theta}^{\overline{1}}\right], \left[\tilde{\Theta}^{\underline{2}}, \tilde{\Theta}^{\overline{2}}\right], \ldots, \left[\tilde{\Theta}^{\underline{P}}, \tilde{\Theta}^{\overline{P}}\right] \right\}, \tag{15}$$

[12]The weight on the different equations can vary, this enables that aggregate conditions have a higher weight than the equations related to agent i to give an example.

[13]The loss function is the mean of the square of the K equations residuals from the Euler equation method. The loss is averaged over the batch size B . The loss function is then: $\Phi^L = \frac{1}{B}\sum^B \frac{1}{K}\sum_{k=1}^K \alpha_k (E_t\left[\Gamma_k^K(s_{t+1,b}, \psi_{t+1,b}, s_{t,b}, \psi_{t,b}, \nu_{t+1,b})\right])^2$, where α_k determines the weight of each equation.

[14]We randomly draw M sets of next period shocks, that is $\{\nu^m_{t+1}\}^M_{m=1}$, to approximate expectations. To increase the efficiency of Monte Carlo integration and reduce the amount of necessary draws, we use the antithetic variate method. The antithetic variates technique creates to a given path $\{\nu_1, \nu_2, \nu_3, \ldots, \nu_{M/2}\}$ also its antithetic path $\{-\nu_1, -\nu_2, -\nu_3, \ldots, -\nu_{M/2}\}$.

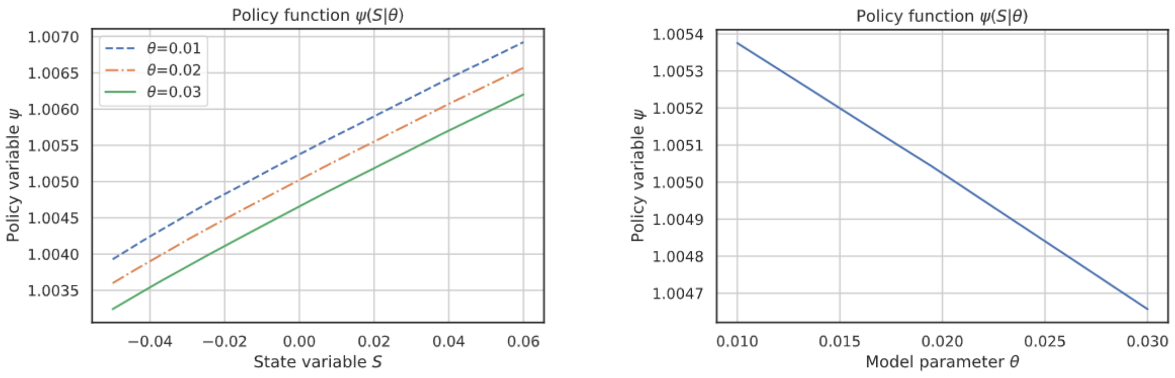

Figure 1: Extended neural network presentation that captures the mapping from state variables as well as model parameters to the policy function. The left plot varies the state variable and displays the policy functions for selected parameters. The right plot fixed the state variables at $S = 0$ and varies the model parameter. The two plots are directly connected as they are created with the same neural network. The points in the left plot at $S = 0.00$ correspond to the policy variable on the right axis for the chosen parameter.

where $\tilde{\Theta}^{\underline{i}}$ and $\tilde{\Theta}^{\bar{i}}$ is the lower bound and upper bound for each parameter that is estimated. Due to the bounded space, we can then draw from a more tight distribution and increase the precision.[15] For each iteration, we draw randomly from the parameter space.[16] Therefore, each economy (batch) has a different parameter combination, which is used to train the extended neural network. After the maximizing step, we redraw the parameter space and then simulate each economy (batch) forward. This also ensures that we train each economy in its relevant stochastic solution domain for the drawn parameter combination. To sum up, we train the neural network over the entire parameter space as we consider in each training step a different set of economies. At the same time, we ensure via simulation that we are solving for the policy function in its relevant stochastic solution domain.

While the extended neural network contains now a solution for the entire parameter space, it is important to validate that the model is solved with a sufficient precision at a given parameter combination. It is well known that economic models may only have a solution for some parameter combinations. Compared to linear models, where the Blanchard Kahn establish local conditions for the existence and uniqueness of the solution, the issue is more complicated with nonlinear models. As this is a general problem for global solution methods, it also directly affects our method. We can use different measures such as the residual error to evaluate the validity of the solution. This helps to evaluate if a solution exist in this area. In particular, we train a neural network that provides a mapping from the parameter space to the residual error. If the residual error is sufficiently small, we keep the solution. Appendix B contains more details on our procedure to validate the solution and disregard parts of the parameter space in an efficient way. We also discuss this issue in more detail in our second proof of concept.

[15]The bounds are conceptually not necessary and the parameters could be drawn from a random distribution.

[16]We draw from a Sobol sequence as it has good distributions in the unit hypercube. We could also draw from other distributions such as a truncated multivariate normal or a distribution motivated by the priors.

Graphical Characterization of the Extended Neural Network A graphical characterization of the extended neural network and its connection to the policy functions can be seen in Figure 1. The left panel shows the mapping from state variables to policy functions for different parameter values. The mapping from state variables to policy variable depends on the chosen model parameter. Importantly, the three mappings are created with the same neural network as the neural network is conditioned on state variables and parameter simultaneously. The right panel further illustrates the idea of treating the model parameters as pseudo state variables. Fixing the state variable(s), the impact of the parameter on the policy function can be directly seen. Importantly, the two plots are created with the same extended neural network. The points in the left plot at $S = 0$ correspond to the policy variable on the right axis for the chosen parameter.

Advantages of the Extended Neural Network Approach The neural network approximates the policy functions for an entire bounded parameter space. As a consequence, we need to solve the neural network only once to evaluate hundreds of thousand different - infinitely many to be precise - parameter combinations! While we theoretically could also adpat more traditional solution methods to incorporate parameters as pseudo state variables, the curse-of-dimensionality prevents this already for slightly complex models. By contrast, a solution approach based on neural networks is much more scalable and can incorporate a large amount of inputs, which allows to relax very significantly the curse-of-dimensionality.

As a consequence, neural networks allow to overcome several key computational isssues in estimating nonlinear models. First, we include each additional parameter as a pseudo state variables, which increases the amount of inputs. Classical methods would be very limited in the amount of extra parameters, whereas this key trick only slightly increases the complexity for the neural network solution method. Second, a model with heterogenous agents consists of hundreds of shocks. Approximating the expectation function for just a few shocks is very cumbersome with Gauss-Hermiture quadrature or a finite state Markov chain, which are usually used for conventional global methods. Exploiting the stochastic setup of our model, we use a Monte Carlo approach for integration that can features hundreds of shocks. Another advantage is that we solve for the entire equilibrium without distinguishing between variables that affect idiosyncratic and aggregate dynamics. Therefore, our approach does not impose any restrictions on the parameters selected for the estimation and we can consider parameters that affect the stochastic steady state. Finally, elevated models are often hard to solve so that tractable approximations are studied and used for an empirical analysis. Our method is very general and covers a large class of macroeconomic models. As a consequence, researchers can employ more complex nonlinear models for their empirical analysis.

2.3 Likelihood, Particle Filter and Neural Networks

The next step is to compute the likelihood, which evaluate the fit of the model with the data in an efficient manner. The nonlinear model solution requires to use a nonlinear filter such as the particle filter. However, the execution of the particle filter is computationally much more costly than the Kalman filter, which can be used in linear setups. This limits the amount of parameter combinations that the estimation procedure can evaluate in due time.

To overcome this issue, we train a neural network that provides a direct mapping from the parameter combination to the likelihood value obtained by the particle filter. This is a so called surrogate model, which is trained on some data points and then provides in a computationally cheap way the outcome of interest. Instead of repeatedly applying the particle filter for each draw, we use this neural network to approximate the results of the particle filter. Specifically, we calculate the likelihoods with the particle filter for different parameter combinations over the parameter space and then use these as data points to train the neural network. The advantage is that we need to run the particle filter much less often in this setup, while we can at the same time evaluate the likelihood over the entire parameter space. We denote this as a neural network based particle filter.

Measurement Equation The measurement equation that connects the state variables with the observables $\mathbb{Y}_t$ can be written as:

$$\mathbb{Y}_t = g(\mathbb{S}_t|\tilde{\Theta}) + u_t, \tag{16}$$

where g is a function and u_t is a measurement error.

Likelihood Evaluation and Particle Filter We use a particle filter to extract the hidden states and shocks due to the nonlinearity of the solution.[17] The particle filter gives us then the likelihood of the model:

$$\mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right). \tag{17}$$

While the particle filter could be directly used within our approach, it can be very time consuming for sufficient complex models. This is a particular problem as the filtering step is usually repeated hundreds of thousand times, which would render estimation infeasible.

Neural Network as Surrogate Model for the Likelihood To overcome this bottleneck, we propose a new neural network-based particle filter method. The particle filter determines the likelihood for the parameters $\tilde{\Theta}$ conditional on the data, which can be written as:

$$\mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) = \Omega^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right), \tag{18}$$

where $\mathcal{L}$ is the likelihood and the function Ω^{PF} is unknown.[18] A normal estimation procedure calculates the value of the likelihood at each drawn point. This implies that $\Omega^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right)$ is

[17]The particle filter (e.g. Herbst and Schorfheide, 2015) 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). However, the particle filter has not been applied in the context of large HANK models so far.

[18]Importantly, the extended neural network is used during the particle filter to calculate the likelihood.

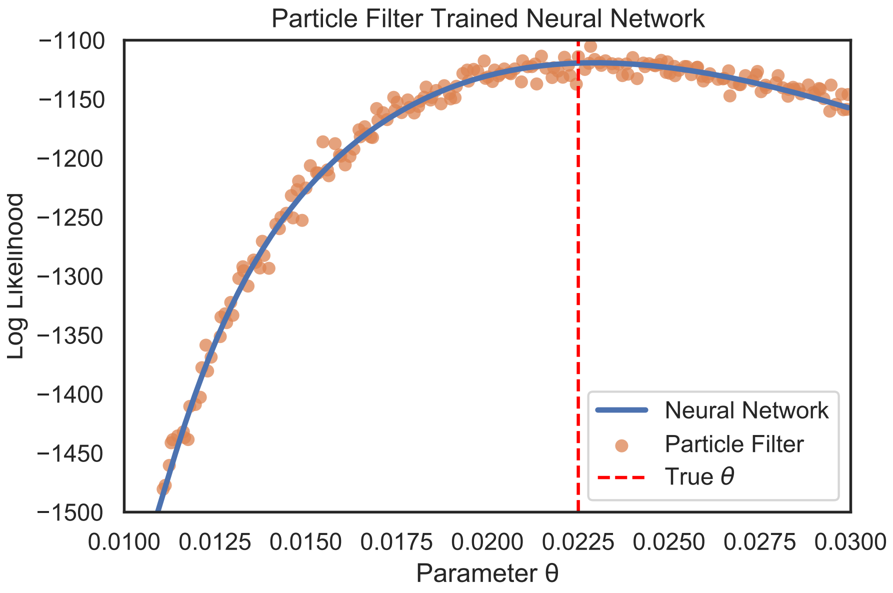

Figure 2: Neural network based particle filter method that captures the mapping from the parameter to the log likelihood. The orange dots represent the data sample, where the log likelihood value has been calculated with the particle filter. The blue line is the neural network, which was trained with these data points from the particle filter. The red dashed line indicates the true value.

evaluated each single time for a given new draw of a parameter. Our strategy differs fundamentally and uses the advantages of neural networks as a very flexible function approximator. We train a neural network Ω_{NN}^{PF} that gives us directly the output of the particle filter:

$$\mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) = \Omega_{NN}^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \tag{19}$$

where $\mathcal{L}$ is the likelihood of the model and Ω_{NN}^{PF} is the neural network associated with the particle filter. This neural network provides the likelihood in a very efficient manner.

Training of the Neural Network Based Particle Filter To train this separate neural network, we create a dataset of parameter values and corresponding likelihoods that we obtained by employing the particle filter. To avoid overfitting of the neural network, we split the calculated points in in a training and testing sample.[19] After we have trained $\Omega_{NN}^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right)$, the likelihood of the model can be evaluated at a specific draw for negligible costs. This neural network is then a surrogate model because it maps the parameters as inputs directly to the associated likelihood value.

Graphical Characterization An example of the neural network based particle filter method can be seen in Figure 2. The orange dots represent the data sample, where the log likelihood value has been calculated with the particle filter. We use these points to train a neural network that directly maps the parameter values into a log likelihood value.

Remarks on the Neural Network Based Particle Filter This approach differs on one important margin to the standard approach of using filters. The standard approach evaluates

[19]Overfitting is not an issue for the policy functions neural networks as we draw always new random points.

the likelihood with the filter without using any information of the likelihood of points that are close to it. While we evaluate the likelihood at only several thousand points, we use the neural network to learn the connection between these points. Conceptually, this idea could be seen as a nonlinear interpolation of the likelihood values with neural networks. This allows us then to evaluate the likelihood at points that we did not assess initially. Once the neural network is trained, we can evaluate the likelihood for millions of parameter combinations in a very short time. Another important advantage of this method is that the neural network removes some of the noise associated with the particle filter as Figure 2 highlights. While other class of models than neural networks could be also used as a surrogate models, we choose neural networks as they are flexible and can approximate nonlinearities.

2.4 Estimation

We can now proceed to the final estimation step.. Equipped with the particle filter trained neural network, we can evaluate each likelihood over the entire parameter space in less than 1ms. This opens up the possibility to either conduct maximum likelihood estimation or Bayesian estimation with millions of draws.

Maximum Likelihood Estimation Maximum likelihood estimation maximizes the likelihood of the model:

$$\tilde{\Theta}^{ML} = \arg\max_{\tilde{\Theta}} \mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \tag{20}$$

We restrict the parameter space to be inside the boundaries of our extended 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 extended neural network covers a large enough area so that the mode lies very likely inside its bounds.

Bayesian Estimation The particle filter similarly allows us to estimate the model with Bayesian methods. Bayesian inference uses the posterior distribution $p\left(\tilde{\Theta}|\mathbb{Y}_{1:T}\right)$, which combines the likelihood with a prior distribution:

$$p\left(\tilde{\Theta}|\mathbb{Y}_{1:T}\right) \propto \mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \times p(\tilde{\Theta}). \tag{21}$$

where $p(\tilde{\Theta})$ is the prior distribution. 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.

We can then construct the posterior distribution with a Random Walk Metropolis Hastings algorithm. We start to draw from the proposal density and can then use our precomputed particle filter neural network to directly evaluate the posterior.

Other Estimation Approaches Other methods such as method of moments, generalized method of moments or impulse response matching could be also decoded in this framework.

Instead of creating the particle filter neural network, the appropriate alternative network would then be trained with selected simulated moments or the impulse response functions.

2.5 Algorithm

The neural network based estimation approach consists of three key steps: i) Train the extended neural network to get the policy functions over the parameter space, 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. A detailed description of the algorithm to estimate macroeconomic models with Bayesian methods is in the Appendix C. Our accompanied codes rely on *PyTorch* as the machine learning framework and Adam (Kingma and Ba, 2014) as the stochastic optimizer.

3 Proofs of Concept

We provide two proofs of concept to establish the validity of our estimation approach. Specifically, we compare our neural network based method to i) the true solution of a small linearized NK model that can be solved analytically and to ii) the estimation of a RANK model with a zero lower bound based on conventional methods for nonlinear models.

3.1 Comparison to an Analytically Derived True Solution

We begin with solving a version of the linearized 3 equation New Keynesian model. The reason for this choice is that this workhorse model has an analytical solution. We can then compare the outcome of the extended neural network with the true solution. We demonstrate that the extended neural network provides a very precise solution for the entire considered parameter space.

The model is a small off-the-shelf NK model with a TFP shock that can be written in linearized form as follows:

$$\hat{X} = E_t\hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi\hat{\Pi}_t + \phi_Y\hat{X}_t - E_t\hat{\Pi}_{t+1} - \hat{R}^F_t\right) \tag{22}$$

$$,\hat{\Pi}_t = \kappa\hat{X}_t + \beta E_t\hat{\Pi}_{t+1}, \tag{23}$$

$$\hat{R}^F_t = \rho_A\hat{R}^F_{t-1} + \sigma(\rho_A - 1)\omega\sigma_A\epsilon^A_t, \tag{24}$$

where the output gap is defined as $\hat{X}_t = (X_t - X)/X$, inflation as $\hat{\Pi}_t = \Pi_t - \Pi$. The strucutral parameters are described in Table 1. Furthermore, the distribution of the shock is $\epsilon^A_t \sim N(0,1)$ and we define $\omega = (1+\eta)/(\eta+\sigma)$ as well as $\kappa = (1-\phi)(1-\phi\beta)(\sigma+\eta)/\phi$. The system of equations can be either solved numerically or analytically.

The analytical solution, which can be derived with the method of undetermined coefficients, is given as:

$$\hat{X}_t = \frac{1-\beta\rho_A}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\hat{R}^F_t \tag{25}$$

	Parameters	LB	UB		Parameters	LB	UB
β	Discount factor	0.95	0.99	θ_Π	MP inflation response	1.25	2.5
σ	Relative risk aversion	1	3	θ_Y	MP 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: The panel shows the parameters of the three equation NK model. We solve the neural network for the parameter space that is spanned by the lower bound (LB) and upper bound (UB) of all parameters.

$$\hat{\Pi}_t = \frac{\kappa}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\hat{R}^F_t \tag{26}$$

The analytical solution points out the idea of extended neural network, in which the solution is simultaneously conditioned on the parameters and state variables. The analytical solution of the output gap and inflation depends on the state variable $\hat{R}^F_t$. At the same time, the output and inflation level also depends on the parameters.

Extended Neural Network To obtain a numerical solution, we use our neural network approach. A neural network is trained to find the policy functions of inflation and output gap. Importantly, the policy functions are simultaneously conditioned on the state variable and the parameters. We minimize the residual error in the equations (22) and (23), while the law of motion of the exogenous state variable is described in equation (24). Table 1 describes the upper and lower bounds of the parameters, for which we solve the extended neural network. While each parameter is varied for demonstration purposes, the method allows also to fix a subset of parameters. Appendix F shows how this model can be mapped in the general form that is outlined in Section 2.

We use 100,000 iterations to train the extended neural network.[20] After each iteration, the economy is simulated for 20 periods to get a new draw for the state variable. While we initially (for the first 5,000 iterations) redraw the parameters after 20 iterations, we change this to a new draw after each iteration afterwards. The batch size is set to 500, which can loosely be thought of having 500 different parallel economies in each iteration to train the neutral network.[21]

The convergence of the neural network is shown in Figure 3. The graph shows the mean weighted residual error. The mean is taken across the batches and the weights of the Euler equation and NKPC are equalized. The residual error drops form an initial level of around 10^{-4} to 10^{-8} on average, which is a decrease of 99.99%. This chart points out that the neural network is very successful in minimizing the residual error over the entire set of parameters.

[20]The neural network contains 5 hidden layers with 128 neurons each. The activation function for the hidden layers are Sigmoid Linear Unit (SiLU). The learning rate of the deep learning algorithm is lowered after 50000, 60000, 70000, 80000 and 90000 iterations.

[21]Our algorithm follows the general setup that we have derived in Section 2.2. The simple setting would actually allow to fine-tune the method. For instance, we could directly sample from our state space as the only state is an exogenous shock. We also do not need Monte Carlo integration due to the linear setup. The expectations would be also pinned down if we would just assume that next the period shock is zero.

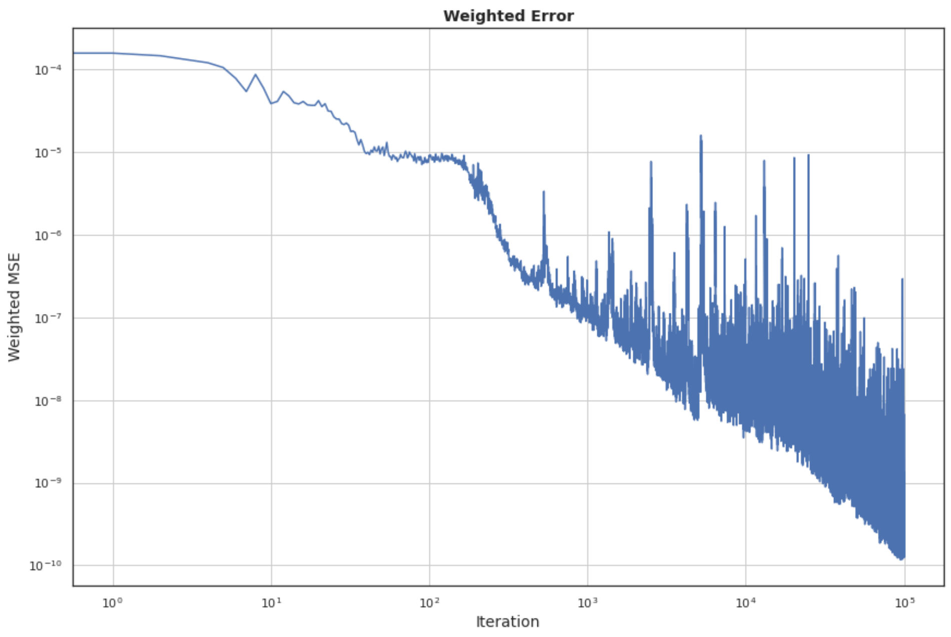

Figure 3: Figure shows the convergence of the neural network over the 100,000 iterations. The line displays the mean weighted residual error. The mean is taken across the batches and the weights of the Euler equation and NKPC are equalized. Both axis use a base-10 logarithmic scale.

Results Figure 4 shows the policy function of the output gap for variations of the parameters. The policy function is evaluated at the one standard deviation of the ergodic distribution and the unvaried parameters are fixed in the middle of their bound. The advantage of the simple model is that the neural network solution can be compared to the true solution. The comparison demonstrates that our extended neural network can capture the true solution as the lines almost perfectly coincide. The shown connection is also highly nonlinear, which emphasizes the potential of neural networks to capture nonlinearities.[22] The variations in the standard deviation of the shock is a noteworthy case. First, both lines are horizontal. The reason is that the model is linearized so that the standard deviation of the shock has no impact on the mapping from R_t^f to $\hat{\Pi}_t$ and $\hat{X}_t$. Second, there is a small uptick in the neural network for low values of the standard deviations. The reason is that we evaluate the neural network at a rather large deviations from the steady state. The stochastic state space visits this region very rarely so that the precision is slightly lower. As the solution is too far off from the normal region, this is not a problem. However, increasing the number of iterations or oversampling large shocks can help to remedy this problem. While we have focused here only on the output gap, Figure 9 in the Appendix demonstrates the same take-aways for inflation. The neural network based solution coincides almost perfectly with the true solution. This concludes the first proof of concept.

[22]While the model is linearized, variations in the structural parameters can be nonlinear.

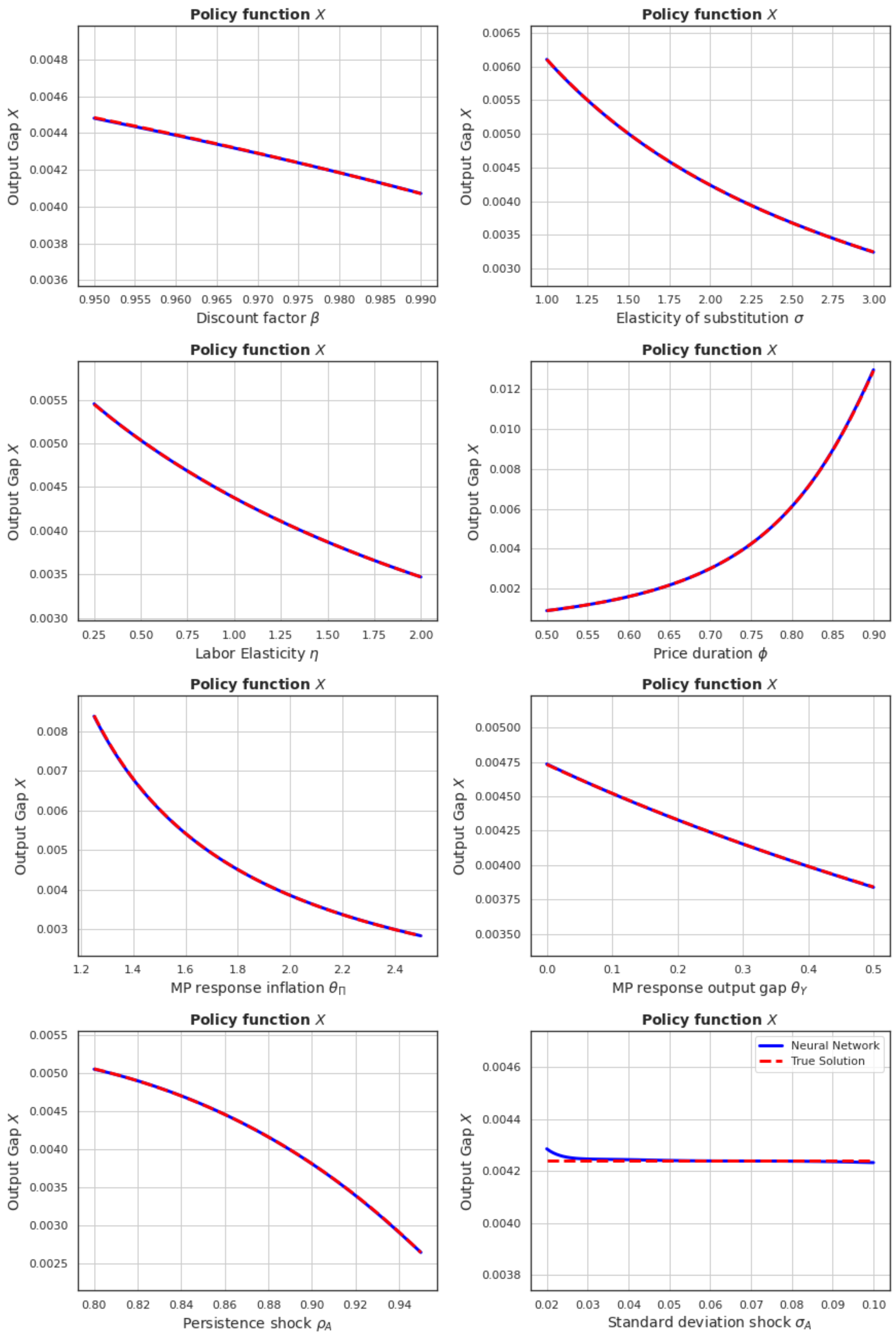

Figure 4: Comparison between the neural network based solution and the true analytical solution. The plot shows how variations in the structural parameter affect the policy function for the output gap $\hat{X}_t$. The policy function is evaluated at the one standard deviation of the ergodic distribution and the unvaried parameters are fixed at their mean.

3.2 Comparison to a Conventional Estimation of Nonlinear Models

We now evaluate if our estimation strategy can recover the true data generating process of a nonlinear model. In particular, we employ a Bayesian estimation with our neural network for a nonlinear RANK model augmented with a ZLB. Additionally, we compare the results to a conventional estimation approach of nonlinear macroeconomic models. The conventional method relies on solving the model with global methods and then evaluating the likelihood with a particle filter for each single draw of the Metropolis Hastings algorithm (Herbst and Schorfheide, 2015). However, the conventional approach restricts the size of the model because of the curse-of-dimensionality and the costs to run the particle filter sufficiently often. Therefore, we focus on a small-scale version of the RANK model with a ZLB.[23] In fact, we can later extend it to our nonlinear HANK model that we are actually after. Another advantage is that this model does not feature an actual solution if the ZLB binds too often, as shown e.g. in Bianchi et al. (2021). This provides a great testing ground how we can assess the validity of a solution at a specific parameter combination.

3.2.1 Model

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

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

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

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:

$$C_t + B_t = W_t H_t + \frac{R_{t-1}}{\Pi_t} B_{t-1} - T_t + Div_t, \tag{27}$$

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

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{\zeta_{t+1}}{\zeta_t} \right) \left(\frac{C_t}{C_{t+1}} \right)^{\sigma} \frac{1}{\Pi_{t+1}} \right], \tag{28}$$

$$\chi (H_t^i)^\eta = (C_t)^{-\sigma} s_t^i W_t. \tag{29}$$

Firms The firm sector consists of a continuum of final goods producer and intermediate goods firms. The final goods retailers buy the intermediate goods and transform it into a

[23]Even though there are impressive examples that estimate nonlinear RANK models (e.g. Gust et al., 2017; Atkinson et al., 2020), the scope of the models is unavoidably still quite limited.

homogeneous final good using a CES production technology:

$$Y_t = \left(\int_0^1 (Y_t^j)^{\frac{\epsilon-1}{\epsilon}} df \right)^{\frac{\epsilon}{\epsilon-1}}, \tag{30}$$

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:

$$P_t = \left(\int_0^1 (P_t^j)^{1-\epsilon)} \right)^{\frac{1}{1-\epsilon}}, Y_t^j = \left(\frac{P_t^j}{P_t} \right)^{-\epsilon} Y_t. \tag{31}$$

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:

$$Y_t^j = ZN_t^j, \tag{32}$$

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

$$W_t = Z_t MC_t. \tag{33}$$

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:

$$\max_{P_t^j} P_t^j \left(\frac{P_t^j}{P_t} \right)^{-\epsilon} \frac{Y_t}{P_t} - MC_t \left(\frac{P_t^j}{P_t} \right)^{-\epsilon} - \frac{\varphi}{2} \left(\frac{P_t^j}{\Pi P_{t-1}^j} - 1 \right)^2 Y_t, \tag{34}$$

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:

$$\left[\varphi^R \left(\frac{\Pi_t}{\Pi} - 1 \right) \frac{\Pi_t}{\Pi} \right] = (1-\epsilon) + \epsilon MC_t + \beta \varphi^R \mathbb{E}_t \frac{\Pi_{t+1}}{R_t} \left[\left(\frac{\Pi_{t+1}}{\Pi} - 1 \right) \frac{\Pi_{t+1}}{\Pi} \frac{Y_{t+1}}{Y_t} \right], \tag{35}$$

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

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 the interest rate response is smoothed with the previous period interest rate. The zero lower bound restricts the nominal interest rate. The rule is given as:

$$R_t = \max \left[1, R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_\Pi} \left(\frac{Y_t}{Y} \right)^{\theta_Y} \right]. \tag{36}$$

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

$$D = \frac{R_{t-1}}{\Pi_t} D_t - T_t. \tag{37}$$

3.2.2 Calibration and Data-Generating-Process

The calibrated model is used as data-generating-process. This provides a controlled environment for our experiment if the neural network can recover the true value of the parameters. The upper panel of Table 2 summarizes the calibration of the model. We set the discount factor β to 0.9975, which implies an annualized real interest rate of 1%. The persistence of the shock is set to 0.7, while the standard deviation of the shock is set to 0.02. This ensures that the model occasionally encounters the ZLB. The remaining parameters are standard.

3.2.3 Estimation

We now estimate the nonlinear model in a Bayesian setup by employing our developed approach, which relies on the extended neural network and the neural network based particle filter. We compare the results to a estimation with a conventional approach.

Estimated Parameters and Priors The estimation includes five structural parameters: The response of the monetary authority to inflation θ_Π and to output θ_Y, the Rotemberg pricing parameter φ as measure of price stickiness as well as the persistence ρ_ζ and standard deviation σ^ζ of the preference shocks. The prior distributions are truncated normal densities.[24] The prior mean correspond to the true value, while the standard deviation σ is very loose to avoid that the results are driven by the prior. The truncation ensures that the drawn parameters lie inside the bounds that have been imposed while solving the extended neural network.[25] The lower panel of Table 2 summarizes the priors.

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 period of 1000 periods. The measurement equation is given as:

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 100 \ln\left(\frac{Y_t}{Y_{t-1}}\right) \\ 400 \ln(\Pi_t) \\ 400 \ln(R_t) \end{bmatrix} + u_t, \tag{38}$$

where the measurement error follows a Gaussian distribution $u_t \sim \mathcal{N}(0, \Sigma_u)$. We include a measurement error to avoid a degeneracy of the particle filter and to include more data series than shocks. As in Gust et al. (2017), the variance of the measurement error for each time

[24]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 mean independent of the bounds.

[25]Even though a truncated prior density is helpful, it is not necessary. The extended neural network can be solved over a distribution without bounds and can, to some extent, extrapolate.

Calibration for the data-generating process					
Parameters		Value	Parameters		Value
β	Discount factor	0.9975	θ_Π	MP inflation response	2
σ	Relative risk aversion	1	θ_Y	MP 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			Conventional 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 2: 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 a conventional approach. The prior type indicate the prior density function, where Trc.N stands for a truncated normal distribution.

series is a fraction m_E of its own variance. We set $m_E = 0.1$. To be consistent, we combine our simulated data series with a measurement error.

Neural Networks Based Estimation Approach The neural network based estimation approach consists of three steps: i) Train the extended neural network to get the policy functions over the parameter space and assess the residual error, ii) Train the neural network based particle filter to get a direct mapping for the likelihood, iii) Run the Metropolis Hastings algorithm.

The computationally most challenging step ist the first one, where we solve for the policy functions. The neural network minimize the residual error in the Euler equation and the NKPC. The neural network provides the policy functions for labor and consumption and is conditioned on the parameters to be estimated and the state variables. Appendix F shows how this model can be expressed in the general form that is outlined in Section 2.

We use 100,000 iterations to train the extended neural network.[26] After each iteration, the economy is simulated for 20 periods. The batch size is set to 500. As an additional element, we train a surrogate model that evaluates the residual error over the entire parameter space. This additional neural network, which helps to evaluate the existence of an equilibrium, is also described in detail in Appendix B. We generate 15,000 likelihoods with the particle filter to train the neural network based particle filter. The surrogate models for the validation and the particle filter are computationally easier so that we choose a less complex neural networks. We now easily obtain the posterior with a Random Walk Metropolis Hastings

[26]The neural networks contains 5 hidden layers with 128 neurons each. The activation function for the hidden layers are parametric Rectified Linear Unit (PReLU).

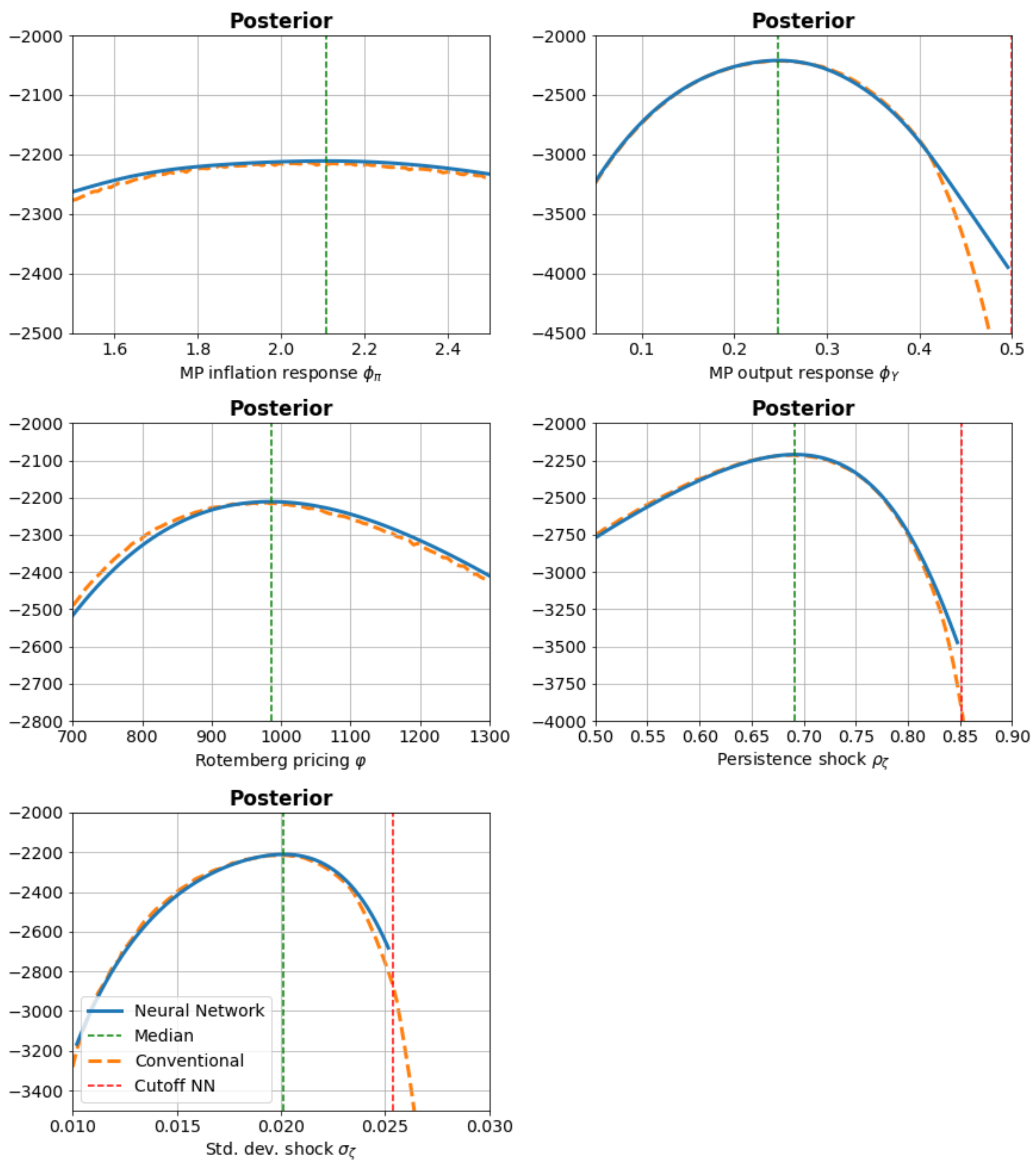

Figure 5: Posterior comparison between the neural network method (solid blue) and the conventional approach (dashed orange). Each estimated parameter is varied and its impact on the posterior shown. The other parameters are fixed at the posterior median. The cutoff value indicates the range for which a sufficient precise solution cannot be found due to deflationary spirals.

algorithm, which uses 50,000 draws after a burn-in.

We compare the results to a conventional nonlinear estimation approach that does not use machine learning techniques to solve the model and calculate the posterior. 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). Appendix D.2 provides more details on the conventional etsimation procedure. The conventional approach also uses the nonlinear model as the data-generating-process.[27] However, this approach is

[27]We use the same sequence of structural shocks and measurement error shocks to generate the data. For

much slower as we need to resolve the model and run the particle filter for each draw. These limitations rationalizes our restriction to a rather low amount of only 50,000 draws in the Metropolis Hastings algorithm.

3.2.4 Results

The estimation results are summarized in Table 2. First of all, the neural network approach performs very well in recovering the true data-generating process. The posterior median is very close to the true value and is always cointained inside the 90% confidence interval. The largest relative deviation from the true value is for the response of inflation, where the posterior median is 2.11 relative to the true value of 2.0. However, the true value is contained in the 90% confidence interval. Furthermore, the posterior median results for the conventional method are very similar. In addition to this, the ranges of the 90% confidence interval are also close.

Figure 5 compares the posterior of the neural network method to the conventional approach. Starting from the posterior median, each estimated parameter is varied and its impact on the posterior shown. This shows that the posterior median is well identified. While the posterior median of inflation is slightly away from the true value, this is related to the rather flat posterior. A longer time series or a lower measurement error would help to identify this parameter more precisely. Both methods provide a similiar shape of the posterior. This shows that solving the neural network and training a surrogate model for the particle filter allows to estimate such models. Nevertheless, Figure 5 also shows that the right tail differs for the parameters θ_Y, σ_ζ and ρ_ζ to some extent. This is related to the impact of the zero lower bound on the solution. If the persistence is high, the standard deviation is large or the monetary authority responds strongly to output, the economy encounters the ZLB very often. This creates a strong deflationary pressure and bias. However, there is a slight difference between the neural network solution method and a conventional global method regarding the deflationary bias. The neural network suggests a slightly lower deflationary bias. As a consequence, the solution is more close to the data-generating process, which explains the higher likelihood value at the right tail for the neural network method.

Furthermore, the deflationary pressure can also result in the non-existence of equilibria, as shown in Bianchi et al. (2021). While this results in a collapse of the algorithm for a conventional global solution method, we do not observe necessarily the same with the extended neural network in the relevant parameter space. Instead, we observe that the residual error is sufficiently larger in this area. In other words, the loss function, which is minimized in the training, is larger in some areas of the parameter space. The reason is that a collapse of the algorithm leads to a breakdown of the economy and ultimately to a very large residual error. In contrast to this, a wedge in the Euler equation can break the deflationary spiral resulting from the risk that the ZLB constraint will become binding. This is exactly the

the neural network, we feed the shocks in the model at the true parameters solved with neural networks. For the conventional method, we feed the shocks in the model at the true parameters solved with standard global methods.

mechanism that we implicitly observe. The neural network allows for a larger residual error to generate a wedge that avoids a deflationary spiral. Therefore, we discard the solutions, where the residual error is larger.[28] To find this area in an efficient manner, we rely on neural networks. We train a surrogate neural network model that provides a mapping from the parameter space to the average residual error. Appendix B contains a general description on training a surrogate neural network for the residual error to disregard parts of the parameter space in an efficient way. We use this surrogate model to discard solutions and only consider solutions with a sufficient low residual error in the estimation. Figure 5 shows the cut-off value if we vary the one parameter. Importantly, the cut-off value is also rather close to the conventional solution approach. Appendix G.2 contains more information on the surrogate model and provides a heat map of the residual error.

4 Estimating a Nonlinear HANK Model

To demonstrate the potential of our developed neural network based estimation approach, we solve and estimate a nonlinear HANK model that features hundreds of state variables, structural shocks and policy variables as well as nonlinearities at the aggregate and individual level. Using this model as our laboratory true-data generating process, we apply our neural network based Bayesian estimation approach to recover the true model. We use our estimation to shed light on how well the parameters related to the heterogeneous setup can be identified using standard aggregate data. For this analysis, we exploit that our approach does not impose any restrictions on the set of parameters that can be included in the estimation. Furthermore, we also emphasize the importance of the connection between idiosyncratic and aggregate risk, which underlines the necessity of establishing such a nonlinear approach.

4.1 Model

The model is a medium scale nonlinear HANK model that captures idiosyncratic and aggregate risk. The first key ingredient is heterogeneity. The households face idiosyncratic income risk and a borrowing limit. The second key ingredient is that the zero lower bound constrains monetary policy. The model features demand, supply and monetary policy shocks. Backward looking components in the form of habit formation and persistence in the monetary policy rule are also included.

[28]This approach to detect areas with a deflationary spiral is only possible because we work with an extended neural network. If we would solve the neural network only for one parameter combination, we could not assess how the residual error for this combination fares compared to other regions. Therefore, the discussed problem can affect other neural network approaches even more severely and is potentially undetectable.

4.1.1 Households

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

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

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 the parameter h controls the degree of habit formation.[29] The budget constraint in real terms can be written as:

$$C_t^i + B_t^i = W_t s_t^i H_t^i + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i - T_t^i + Div_t^i, \tag{39}$$

where Div_t^i is the real dividend, W_t is real wage, H_t^i is labor, R_t is the gross nominal interest rate, Π_t is the gross inflation rate and T_t^i is real lump sum taxes and. The agents individual labor productivity s_t^i is stochastic and follows an AR(1) process in logs $s_t^i = \rho_s s_{t-1}^i + \epsilon_t^{s,i}$. The agents face a borrowing limit $\underline{B}$, which implies:

$$B_t \geq \underline{B}. \tag{40}$$

The first order conditions can be written as

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{\zeta_{t+1}}{\zeta_t} \right) \left(\frac{\lambda_t^i}{\lambda_{t+1}^i} \right)^{\sigma} \frac{1}{\Pi_{t+1}} \right] + \mu_t^i, \tag{41}$$

$$\lambda_t^i = C_t^i - hC_{t-1} \tag{42}$$

$$\chi (H_t^i)^\eta = (\lambda_t^i)^{-\sigma} s_t^i W_t \tag{43}$$

where $\mu_t^i \geq 0$ is the normalized multiplier on the individual borrowing limit in equation (40).

4.1.2 Firms

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

Final Goods Producers The final goods retailers buy the intermediate goods and transform it into the homogeneous final good using a CES production technology:

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

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:

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

[29] Auclert et al. (2020) discuss the role of habit formation for the marginal propensity to consume.

Intermediate Goods Producers 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:

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

where Z_t is the total factor productivity. Total factor productivity follows a stochastic trend

$$Z_t = g_t Z_{t-1}, \tag{47}$$

where the trend growth rate is subject to idiosyncratic shocks

$$g_t = \bar{g} \exp(\epsilon_t^g). \tag{48}$$

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

$$W_t = Z_t MC_t. \tag{49}$$

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:

$$\max_{P_t^j} P_t^j \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} \frac{Y_t}{P_t} - MC_t \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} - \frac{\varphi}{2}\left(\frac{P_t^j}{\Pi P_{t-1}^j} - 1\right)^2 Y_t, \tag{50}$$

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:

$$\left[\varphi^R \left(\frac{\Pi_t}{\Pi} - 1\right)\frac{\Pi_t}{\Pi}\right] = (1-\epsilon) + \epsilon MC_t + \beta \varphi^R \mathbb{E}_t \frac{\Pi_{t+1}}{R_t}\left[\left(\frac{\Pi_{t+1}}{\Pi} - 1\right)\frac{\Pi_{t+1}}{\Pi}\frac{Y_{t+1}}{Y_t}\right], \tag{51}$$

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

$$Div_t = Y_t - W_t Y_t. \tag{52}$$

The dividends are distributed equally among the households so that $Div_t = Div_t^i$.[30]

4.1.3 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 the interest rate response is smoothed with the previous period interest rate. In addition to this, there are i.i.d. monetary policy shocks mp_t. The zero lower bound restricts the nominal

[30]An alternative formulation would be to make the dividend payments depending on the individual productivity of the agent along the lines of Kaplan et al. (2018).

interest rate. The rule is given as:

$$R_t^N = \left(R_{t-1}^N\right)^{\rho_R} \left(R \left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi} \left(\frac{Y_t}{Z_t Y}\right)^{\theta_Y} \right)^{1-\rho_R} \exp(mp_t), \tag{53}$$

$$R_t = \max\left[1, R_t^N\right]. \tag{54}$$

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

$$D_t = \frac{R_{t-1}}{\Pi_t} D_{t-1} - T_t. \tag{55}$$

4.1.4 Market Clearing

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

$$N_t = \int N_t^j dj = \int s_t^i H_t^i di \tag{56}$$

$$D_t = \int B_t^i di \tag{57}$$

$$Y_t = \int C_t^i di. \tag{58}$$

4.2 Calibration

The calibrated model is used as data-generating-process to provide a laboratory setup for our approach. The upper panel of Table 3 summarizes the calibration of the model. We fit the model to the current low interest rate environment and capture the heterogeneity of households in line with the nonlinear models of Gust et al. (2017) and Bianchi et al. (2021) as well as the heterogeneous agent frameworks of Kaplan et al. (2018) and Fernández-Villaverde et al. (2021). We set the discount factor β to 0.995, which implies a 200 basis points contribution to the annualized real interest rate. The persistence of the preference shock is set to 0.7, while the standard deviation is set to 0.03. This ensures that the economy encounters the zero lower bond regularly. The standard deviation of the growth rate and monetary policy shock are set to be in line with the standard deviation of GDP per capita output growth and the effective federal funds rate.

The persistence of the individual labor productivity shock ρ_s and the individual borrowing limit $\underline{B}$ follows Fernández-Villaverde et al. (2021). The standard deviation of the individual shock targets a 30% share of borrowers to be around 30% in line with Kaplan et al. (2018). Government debt D is set to 0.25. The remaining parameter are standard.

4.3 Estimation

We estimate the nonlinear HANK model with our newly developed neural network based Bayesian methods. In particular, we use our extended neural network to obtain the policy functions and the neural network based particle filter. To provide a controlled environment, the calibrated model is used as the true-data generating process.

Estimated Parameters and Priors The estimation includes 12 structural parameters, which can be roughly separated as parameters related to idiosyncratic risk and aggregate risk. The parameters related to the idiosyncratic risk are the borrowing limit of individual households $\underline{B}$ and the parameters of the stochastic process of the labor productivity ρ_s and σ_s. The remaining parameters are habit h, Rotemberg pricing φ, the persistence of the Taylor ρ_r, the monetary policy response to inflation θ_Π and output θ_Y, the persistence of the preference shock and the standard deviations of the three shocks σ_ζ, σ_g and σ_{mp}. The prior densities are truncated normals for all parameters. 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 extended neural network.

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 period of 500 periods. The measurement equation is given as:

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 100 \ln\left(\frac{Y_t}{Y_{t-1}/g_t}\right) \\ 400 \ln(\Pi_t) \\ 400 \ln(R_t) \end{bmatrix} + u_t, \tag{59}$$

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 $m_E = 0.1$ of its own variance.

Neural Networks Based Estimation Approach The neural network based estimation approach consists of three steps: i) Train the extended neural network to get the policy functions over the parameter space, ii) Train the neural network based particle filter to get a direct mapping for the likelihood, iii) Run the Metropolis Hastings algorithm.

We use two neural networks to train the policy function over the parameter space. The first neural network provides the individual policy functions of labor and the multiplier on the borrowing constraint. This neural network is conditioned on the agent's individual states, the entire state vector, which includes all agents individual states and in the aggregate states, and the estimated parameters. The second neural network provides the aggregate policy functions, namely inflation and the wage, and is conditioned on the state vector and the estimated parameters. These neural networks are jointly trained to minimize the residual error in the individual agents' equations and in aggregate equations. In particular, the loss function includes the Euler equation and a condition if the borrowing limit is binding for each single agent. The aggregate conditions are the New Keynesian Phillips Curve, bond market clearing and product market clearing.[31] We set the amount of agents L to 100. This

[31]An advantage of neural network is that we can include more equations than policy functions in the Euler

implies, we have together 205 state and 12 pseudo state variables and 206 equations, where we minimize the residual error. We use 200,000 iterations to simultaneously train these two extended neural network.[32] After each iteration, the economy is simulated for 20 periods. The batch size is set to 100 and we evaluate the Monte Carlo expectation using 100 draws. Appendix F shows how this model can be expressed in the general form that is outlined in Section 2.

For the second step, we generate 15,000 likelihoods with the particle filter. For this, we randomly draw from the parameter space and then run the particle filter, where we use the extended neural networks as crucial input. We then use these 15,000 data points to train the surrogate neural network that captures the outcome of the particle filter. Specifically, we use 75% of the sample to train the neural network and the remaining fraction for the validation of the neural network.[33] We train this neural network related to the likelihood over 10,000 iterations with a batch size of 100. Once, we have obtained the the neural network based particle filter, we have a direct mapping from the parameter values to the likelihood. Importantly, this mapping is computationally very fast because our approach front-loads the computational costs. Thus, the costs to evaluate a single parameters draw is at this stage now very fast despite estimating a nonlinear HANK model.

Equipped with these objects, we can now move to the Random Walk Metropolis Hastings (RWMH) algorithm. We set the amount of draw to 1 million (after a burn-in) to obtain the posterior distribution. It takes us less than 2 days with a modern day desktop computer to run the described estimation procedure, which consists of training the extended policy functions, the neural network based particle filter and the RWMH algorithm.

4.4 Results

We estimate 12 parameters of the nonlinear HANK model with our neural networks approach. The results are summarized in the lower panel of Table 3. The results show that the posterior median is very close to the true value. In particular, the true value is contained in the 90% credible interval for all parameters. This demonstrates that our method is well suited to estimate complex nonlinear models.

Our neural networks approach has two advantages over more conventional estimation procedures of HANK models. The first advantage is that we can estimate HANK models in its fully nonlinear specification. As a consequence, our method accounts fully for the impact of the zero lower bound (and also the agents' borrowing limit). Other approach usually either restrict the aggregate dynamics (by relying on linearisation or perfect foresight) or the heterogeneous setup (by using RANK or TANK models). To the best of our knowledge, this is the first estimation of a nonlinear HANK model.

equation method. In particular, we have added three aggregate conditions, while conventional methods would only include two conditions.

[32]The neural networks contains 5 hidden layers with 128 neurons each. The activation function for the hidden layers are Mish, which is self-regularized non-monotonic function and defined as $f(x) = x\tanh(softplus(x))$.

[33]The neural network features 128 nodes and four hidden layers. We use SiLU (Sigmoid Linear Unit) as activation function.

Calibration for the data-generating process					
Parameters		Value	Parameters		Value
β	Discount factor	0.995	θ_Π	MP inflation response	2
σ	Relative risk aversion	1	θ_Y	MP output response	0.25
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	ρ_s	Persistence labor productivity	0.8
g	Average growth rate	1.00	σ_s	Std. dev. labor productivity	2.0%
h	Consumption habit	0.1	ρ_ζ	Persistence preference shock	0.7
φ	Rotemberg pricing	1000	σ_ζ	Std. dev. preference shock	2.8%
Π	Inflation target	1.005	σ_g	Std. dev. growth rate shock	0.9%
Y	Output target	1	σ_{mp}	Std. dev. MP Shock	0.3%
ρ_r	Persistence MP rule	0.25			

Estimation								
Par.	Prior					Neural Network		
						Posterior		
	Type	Mean	Std	Lower Bound	Upper Bound	Median	5%	95%
Parameters related to idiosyncratic risk								
$\underline{B}$	Trc.N	-0.15	0.05	-0.5	0.0	-0.15	-0.07	-0.23
ρ_s	Trc.N	0.8	0.01	0.7	0.9	0.80	0.78	0.81
σ_s	Trc.N	2.0%	0.5%	0.01%	3.0%	1.88%	1.70%	2.05%
Parameters related to aggregate risk								
h	Trc.N	0.5	0.01	0.0	0.7	0.50	0.48	0.51
φ	Trc.N	1000	25	700	1300	1000	962	1038
ρ_r	Trc.N	0.5	0.01	0.0	0.7	0.50	0.49	0.52
θ_Π	Trc.N	2.0	0.025	1.5	2.5	2.00	1.96	2.04
θ_Y	Trc.N	0.25	0.025	0.125	0.5	0.26	0.23	0.29
ρ_ζ	Trc.N	0.7	0.025	0.6	0.8	0.71	0.69	0.73
σ_ζ	Trc.N	2.8%	0.25%	0.1%	4.0%	2.77%	2.61%	2.93%
σ_g	Trc.N	0.9%	0.1%	0.01%	1.5%	0.91%	0.83%	1.00%
σ_{mp}	Trc.N	0.3%	0.1%	0.01%	0.6%	0.32%	0.29%	0.35%

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 compares the posterior for the neural network based estimation with a conventional approach. The prior type indicate the prior density function, where Trc.N stands for a truncated normal distribution.

The second advantage is that our method does not restrict the choice of parameters that we can include in the estimation. As the model solves simultaneously for the (stochastic) steady state and the aggregate dynamics, we can include parameters that affect the (stochastic). To highlight this feature, we have included the following parameters related to the idiosyncratic risk: borrowing limit $\underline{B}$, individual labor productivity's persistence ρ_s and standard deviation σ_s. Therefore, our results can show how much aggregate data can help to identify these parameters. The first row of Figure 6 shows the posterior for these parameters. The posterior is rather flat in this area. This indicates that standard aggregate data used for macroeconomic models does not contain too much information for estimating the parameters related to the degree of heterogeneity or inequality. Including distributional data is probably necessary to better pin down such values. At the same time, our results also suggests that an estimation of a HANK model that uses standard aggregate data can rely on a carefully

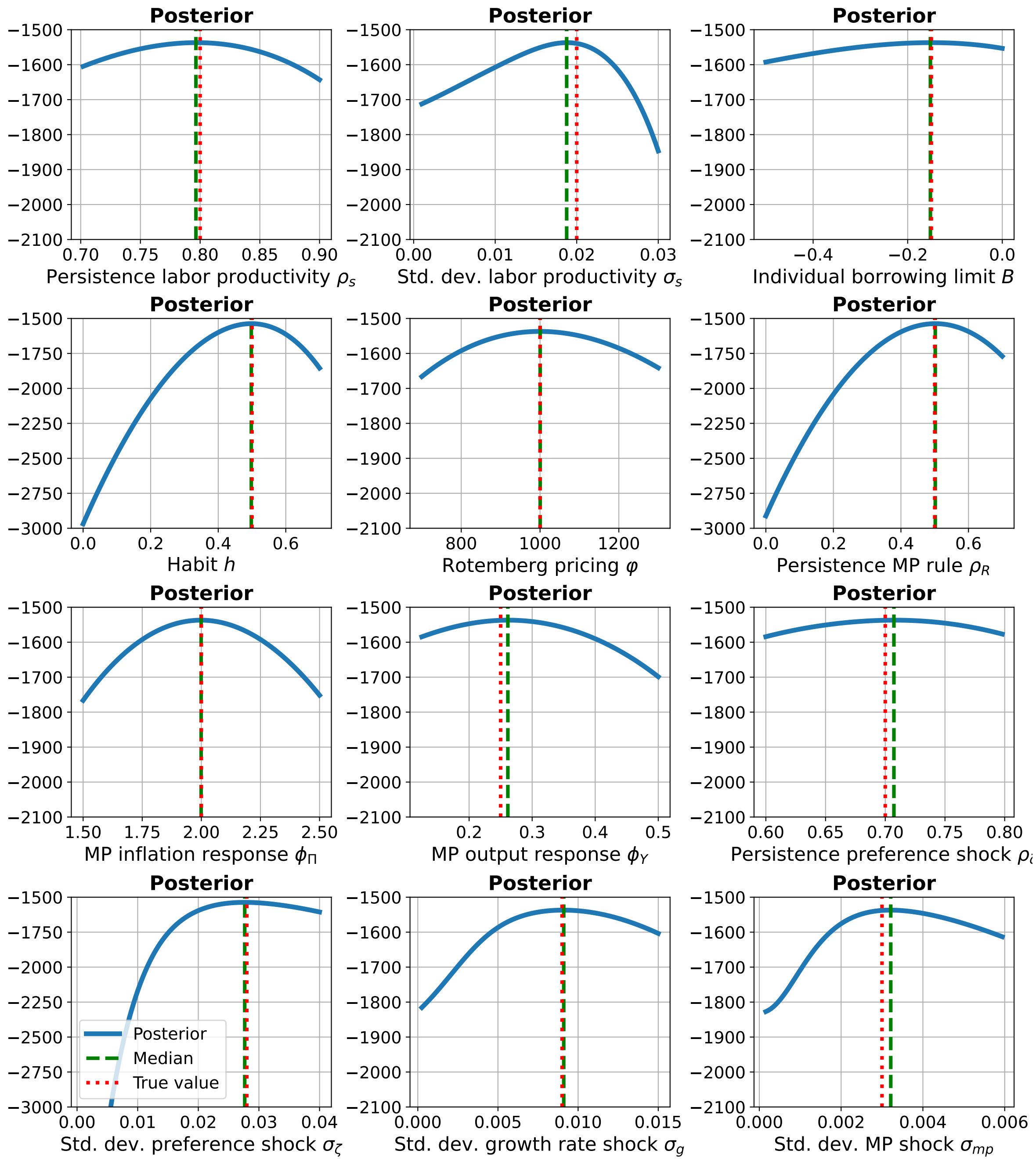

Figure 6: Posterior of the nonlinear HANK model estimated with the developed neural network estimation procedure. Each parameter is varied, while the other parameters are fixed at the posterior median. The posterior median is the green dashed line, while the true value from the data-generating process is the red dotted line.

calibrated version for such parameters.

We also assess the posterior of the remaining 9 parameters, which can be loosely attributed to the aggregate dynamics. We find that the posterior is in most cases rather informative. This is particularly the case for the preference shock, which is our main exogenous driver. We also observe a strong curvature of the posterior for habit and the persistence. Comparing the posterior median to the true value, Figure 6 also highlights that our approach can recover the true-data generating process.

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 an computational 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 agents setup.

We apply our techniques to estimate a nonlinear HANK model, which features idiosyncratic and aggregate nonlinearities simultaneously. Using a laboratory setup, we show that our method can recover the true data generating process. We also provide two proofs of concept for our approach by comparing the neural networks method to an analytical solution and an conventional estimation approach.

The proposed neural network based estimation method opens up 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.

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.

Auclert, A., Rognlie, M., Straub, L., 2020. Micro jumps, macro humps: Monetary policy and business cycles in an estimated HANK model. National Bureau of Economic Research.

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

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.

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.

Chen, H., Didisheim, A., Scheidegger, S., 2021. Deep Structural Estimation: With an Application to Option Pricing. mimeo.

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.

Fernández-Villaverde, J., Hurtado, S., Nuño, G., 2019. Financial frictions and the wealth distribution. Technical Report. National Bureau of Economic Research.

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.

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.

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.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

Krusell, P., Smith, A., 1998. Income and wealth heterogeneity in the macroeconomy. Journal of Political Economy 106, 867–896.

Le Grand, F., Ragot, X., 2021. Managing inequality over business cycles: Optimal policies with heterogeneous agents and aggregate shocks. International Economic Review .

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

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.

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 Deep Learning and Neural Networks

Deep learning is a class of machine learning techniques with (deep) neural networks as fundamental building block.[34] In this paper, we outline a neural network based solution and estimation approach. This strategy can be applied to complex and large macroeconomic models that would have been considered to be out of reach previously. To achieve this task, our approach utilizes two remarkable features of neural networks. The first feature is that neural networks can approximate any continuous function as long as the neural network is sufficient large. This is the so-called universal approximation theorem. The theorem ensures that it is in theory possible to approximate (large) macroeconomic models with neural networks. The second feature is that the neural network can handle large amount of inputs. Specifically, a neural network based solution method is very scalable because additional inputs can be added at rather low computational costs. As a consequence, neural networks can tackle the curse-of-dimensionality so that large models can be captured. However, there is more to the universal approximation theorem and the scalability of neural networks. We demonstrate how to exploit these features of neural neural networks to establish a novel path to the estimation of macroeconomic models.

Before outlining our approach in the next sections, we provide a short-primer on the design and training of neural networks. Afterwards, we discuss the universal approximation theorem and the scalability in more detail.

A.1 Deep Neural Networks

Neural networks are a mathematical function that maps some inputs S into outputs Y:

$$Y = \psi_{NN}(S|W) \tag{60}$$

where $\psi_{NN}(\cdot)$ is the neural network and W are the parameters of the neural network.

The neural network consists of several layers, which can be divided in the input layer, hidden layer(s) and the output layer. The first layer is the input layer, which is visible. Then, the neural network features a number of hidden layers. The final layer is the output layer. Each hidden layer consists of several neurons, which can be seen as nodes in the neural network and are explained in detail in the next paragraph. The amount of neurons determines the width of the layer. The amount of layers determines the depth of the neural network. A neural network with more than 3 layers in total is classified as deep. Figure 7 provides an example of a deep neural network with three inputs, 2 hidden layers with 6 neurons each, and 4 outputs. The displayed layers are dense as each layer is fully connected to each input from the previous layer.

The neural network composes mathematical functions that are performed at the single neurons in the network. A single neuron assigns its inputs $s_1, s_2, \dots s_S$ some weights

[34]Goodfellow et al. (2016) is a very good overview for artificial intelligence and in particular deep learning. Fernández-Villaverde et al. (2020) or Maliar et al. (2021) also discuss machine learning in the context of macroeconomic modeling.

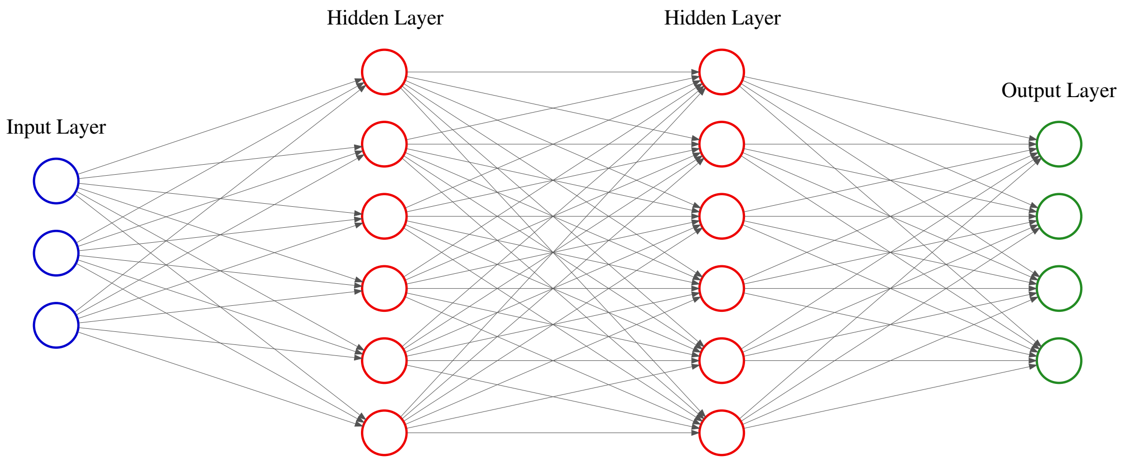

Figure 7: Example of a deep neural network.

$w_1, w_2, \ldots, w_S$ and takes it sum (adjusted by a bias/constant w_0).[35] This value is then taken to a nonlinear activation function $h(\cdot)$,such as ReLU (rectified linear unit) or hyperbolic tangent function, which then results a single output $\tilde{y}$

$$\tilde{y} = h(w_0 + \sum_{i=1}^{S} w_i s_i) \tag{61}$$

The activation function $h(\cdot)$ helps the neural network to approximate nonlinearities in the data. The choice of the activation function for the hidden layers affects how well the neural network can learn the underlying features from the data. The activation function for the output layer also directly determines the possible outcomes of the neural network. For instance, a sigmoid function at the output layer would restrict the potential values to lie between 0 and 1. As shown in Figure 7, these neurons are stacked to form the neural network. The entire neural network can then be summarized as all the parameters at each single neuron, which is the parameter vector W.

Loss Function and Training The next step is to train the parameter vector W. We are interested in minimizing the loss between the actual data and the prediction of the neural network. A popular criteria for continuous variables is for instance the mean squared error loss Φ^L:

$$\Phi^L(W) = 1/B \sum_{i=1}^{B} (Y_i - \psi_{NN}(S_i|W))^2 \tag{62}$$

where Y_i is one data point, $\psi_{NN}(S|W)$ is the prediction of the neural network, which depends on the parameter vector W, and S_i is the input from the data.[36] This loss is evaluated using a total batch B of data points.

[35]Depending on the position of the neuron in network, the input is either directly from the inputs or from the outcome of the previous layer.

[36]There are alternative specifications for a loss function. The mean squared error loss is used as example here because macroeconomic models can be expressed with such a loss function as shown later.

The next step is to optimize the parameters W to minimize the mean squared error loss:

$$W = \arg\min_{W} \Phi^{L}(W) \tag{63}$$

The optimization relies on a iterative stochastic gradient descent method. This updates the parameter vector W until the code converges to a local minima. An important step in finding these weights is backpropagation. This allows to compute how a change in the weights affect the final loss. The step size in updating the weights depends on the learning rate. The setting of the learning rate is important to avoid local minima (sufficient large rate) but also to ensure convergence (sufficient small rate).

One problem with neural networks is overfitting. To avoid this problem, the data is separated usually in a training and test data. It should be emphasized that this is not a problem for macroeconomic models as we can always generate new data with the model to avoid overfitting. In that regard, macroeconomic models provide us with a big data environment, which is very helpful for the use of deep learning techniques.

The training of such a neural network is usually based on graphics processing units (GPUs) as these can be used to parallelize many but rather simple activities. PyTorch and TensorFlow are popular open source machine learning librarys that can be used to build and train neural networks.

A.2 Universal Approximation Theorem

An important argument for the usage of neural networks is the universal approximation theorem, which 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; Bach, 2017).[37] Importantly, macroeconomic models can be casted in such a finite-dimensional space. As a consequence, neural networks can in theory be applied to solve macroeconomic models.

A.3 Scalability

A particular problem in solving models with global methods is the well-known curse-of-dimensionality. Extending the complexity of the model (by raising the number of state variables) results in an exponential increase of the computational problem. As a consequence, it is infeasible to solve large and complex models with such classical solution techniques.

However, neural networks allow to break the curse-of-dimensionality as they can handle high dimensional problems much better than classical function approximators (Barron, 1993; Bach, 2017). The reason is that the number of neurons grows linearly with the number of the dimensions of problem, while in the case of traditional function approximators the size

[37]The theorem relies on some conditions for the activation functions that are used in the hidden layer. It holds for instance for ReLU or the sigmoid function. The neural networks that have been discussed can be classified as feedforward. This implies that there are no loops or cycles in the neural network so that the information only moves forward.

of the problem grows exponentially (Fernández-Villaverde et al., 2020). This scalability of neural networks allows to handle models with a large number of states and tackle the curse-of-dimensionality. As a consequence, neural networks can also in practice be applied to solve complex macroeconomic models.

But, there is more to it than this. We explore how to use the scalibility of neural networks to estimate macroeconomic models. The key trick is to treat the parts (parameters) of the model that we want to estimate as inputs for the neural network. The neural network can then be trained simultaneously not only for one economy but for all possible economies that should be estimated. Importantly, the universal approximation theorem also provides the theoretical groundwork for this approach as we only need a neural network with a sufficient width to include this additional inputs. Exploiting these two features, we are able to outline a general neural network based solution and estimation method.

B Residual Error Neural Network

An important step is to check if the model was solved with a sufficient precision over the parameter space. This is important as there does not necessarily exist a solution at each considered parameter combinations. While in linearized models the Blanchard Kahn conditions directly can control for this, this is not the case in a nonlinear model. As this is a general problem for global solution methods, it also directly affects our method.

To evaluate the solution, 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 extended neural network. The residual error is the weighted mean residual error when simulating the model for a sufficient large amount of time.[38] The residual error depends directly on the parameter: $R(\tilde{\Theta})$. In particular, we are interested in finding a function that directly maps the parameter combination to the residual function:

$$R(\tilde{\Theta}) = \Omega^{RE}(\tilde{\Theta}) \tag{64}$$

where Ω^{RE} is an unknown function.

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

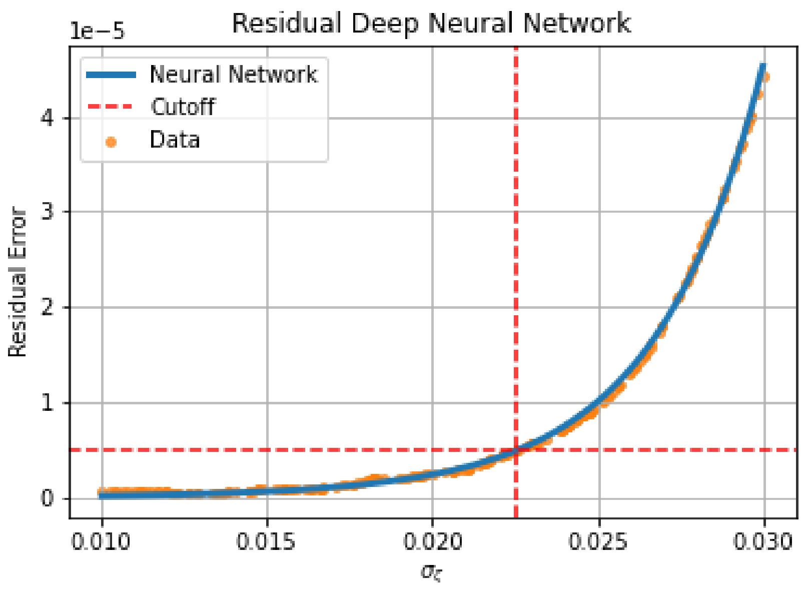

Figure 8: Residual error and surrogate neural network. The orange dots represent the data sample, where the residual error has been calculated. The blue line is the neural network, which was trained with these data points from the residual error. The red dashed lines indicates the cut-off value and parameter values, for which the error is sufficiently small.

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:

$$R(\tilde{\Theta}) = \Omega^{RE}_{NN}(\tilde{\Theta}) \tag{65}$$

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 dataset of parameter values and corresponding mean residual errors.[39] 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.

A graphical characterization of the residual error can be seen in Figure 8. The orange dots represent the data sample, where the residual error has been calculated using the extended

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

neural network in a simulation. We use these points to train a neural network that directly maps the parameter values into a log likelihood value. The graph shows that there is a strong uptick in the residual error if the parameter is further increased. The increase in the mean error is the result that the neural network cannot find an admissible solution. We set a cut-off point, marked as the red line, to decide if a solution is feasible. If the value is below the point, the solution is admitted. Otherwise, we disregard this parameter combination.

C Neural Network-Based Solution Algorithm for HANK

The algorithm uses a neural network approach to approximate policy functions based onMaliar et al. (2021).[40] We are going to use two separate neural networks for the individual and aggregate policy functions. Our neural network

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

   $$\begin{pmatrix} \{N^i_t\}^L_{i=1} \\ \{\lambda^i_t\}^L_{i=1} \end{pmatrix} = \left\{\psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}^L_{i=1} \tag{66}$$

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

   $$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{67}$$

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, so that we start with

   $$\{N^i_t\}^L_{i=1}, \{\lambda^i_t\}^L_{i=1}, \Pi_t, \tilde{W}_t \tag{68}$$

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

   $$\tilde{T}_t = \left(\frac{1}{L}\sum_{i=1}^{L} B^i_{t-1}R_{t-1}\right)\frac{1}{\Pi_t g_t} \tag{69}$$

   $$N_t = \left(\frac{1}{L}\sum_{i=1}^{L} N^i_t s^i_t\right) \tag{70}$$

   $$\tilde{Y}_t = N_t \tag{71}$$

   $$\tilde{\psi}_t = \tilde{W}_t \tag{72}$$

   $$\tilde{Div}_t = \tilde{Y}_t - \tilde{W}_t N_t \tag{73}$$

[40] Maliar et al. (2021) use this approach to solve a consumption saving problem and Krusell-Smith economy

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

$$R_t^N = \left(R_{t-1}^N\right)^{\rho_R} \left(R \left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi} \left(\frac{\tilde{Y}_t}{\tilde{Y}}\right)^{\theta_Y} \right)^{1-\rho_R}, \tag{74}$$

$$R_t = \max\left[1, R_t^N\right] \tag{75}$$

We pursue with calculating for each household individual variables:

$$\left\{ \tilde{C}_t^i = \left[\frac{s_t^i \tilde{W}_t}{\chi (H_t^i)^\eta}\right]^{\frac{1}{\sigma}} \right\}_{i=1}^L \tag{76}$$

$$\left\{ \omega_t^i = \tilde{W}_t s_t^i N_t^i + \frac{B_{t-1}^i R_{t-1}}{\Pi_t} - \tilde{T}_t + \tilde{Div}_t \right\}_{i=1}^L \tag{77}$$

$$\left\{ B_t^i = \omega_t^i - C_t^i \right\}_{i=1}^L \tag{78}$$

Aggregate consumption is given as

$$C_t = \frac{1}{L} \sum_{i=1}^L C_t^i \tag{79}$$

Update the part to Monte Carlo Integration We use the all-in-one expectation method of Maliar et al. (2021), which uses two randomly drawn shocks for each AR(1) process to evaluate the expectations:

$$\text{Random Draws 1:} \left\{\epsilon_t^{\zeta,1}\right\}_{i=1}^L, \epsilon_t^{\zeta,1} \tag{80}$$

$$\text{Random Draws 2:} \left\{\epsilon_t^{\zeta,2}\right\}_{i=1}^L, \epsilon_t^{\zeta,2} \tag{81}$$

We first proceed with random draw 1 to calculate the next period values of the stochastic state variables, that is:

$$\ln(\zeta_{t+1}^1) = \rho_\zeta \ln(\zeta_t) + \epsilon_{t+1}^{\zeta,1} \tag{82}$$

$$\ln(s_{t+1}^1) = \rho_s \ln(s_t) + \epsilon_{t+1}^{s,1} \tag{83}$$

where the superscript indicates to which shock draw the next period value is associated. We can now calculate the individual control variables for the next period:

$$\begin{pmatrix} \{N_{t+1}^{i,1}\}_{i=1}^L \\ \{\lambda_t^{i,1}\}_{i=1}^L \end{pmatrix} = \psi_1\left(\mathbb{S}_{t+1}^1; \Theta\right), \tag{84}$$

and similarly for the aggregate control variables:

$$\begin{pmatrix} \Pi_t^1 \\ \tilde{W}_t^1 \end{pmatrix} = \psi_2\left(\mathbb{S}_{t+1}^1; \Theta\right) \tag{85}$$

We can now calculate the aggregate variables again

$$T^1_{t+1} = \left(\frac{1}{L}\sum_{i=1}^{L} B^i_t R_t\right)\frac{1}{\Pi^1_{t+1}} \tag{86}$$

$$N^1_{t+1} = \left(\frac{1}{L}\sum_{i=1}^{L} N^{i,1}_{t+1} s^{i,1}_{t+1}\right) \tag{87}$$

$$Y^1_{t+1} = AN^1_{t+1} \tag{88}$$

$$Div^1_{t+1} = Y^1_{t+1} - W^1_{t+1}N^1_{t+1} \tag{89}$$

We pursue with calculating for each household individual variables:

$$\left\{C^i_t = \left[\frac{s^{i,1}_{t+1}W^1_{t+1}}{\chi(H^{i,1}_{t+1})^\eta}\right]^{\frac{1}{\sigma}}\right\}_{i=1}^{L} \tag{90}$$

$$\left\{\omega^{i,1}_{t+1} = W^1_{t+1}s^{i,1}_{t+1}N^{i,1}_{t+1} + \frac{B^i_t R_t}{\Pi_{t+1^1}} - T^1_{t+1} + Div^1_{t+1}\right\}_{i=1}^{L} \tag{91}$$

$$\left\{B^{i,1}_{t+1} = \omega^{i,1}_{t+1} - C^{i,1}_{t+1}\right\}_{i=1}^{L} \tag{92}$$

Aggregate consumption is given as

$$C^1_{t+1} = \frac{1}{L}\sum_{i=1}^{L} C^{i,1}_{t+1}. \tag{93}$$

We now calculate the Euler error for the Euler equations, that households satisfy the borrowing limit, the New Keynesian Philipps Curve, the resource constraint in period t and $t+1$, and market clearing for the bond in period t and $t+1$.

$$\left\{R^{i,1}_1 = \beta R_t\left[\left(\frac{\zeta^1_{t+1}}{\zeta_t}\right)\left(\frac{C^i_t}{C^{i,1}_{t+1}}\right)^\sigma \frac{1}{\Pi^1_{t+1}}\right] - \lambda^i_t\right\}_{i=1}^{L} \tag{94}$$

$$\left\{R^{i,1}_2 = \Psi^{FB}\left(B^i_t + \underline{B}, 1-\lambda^i_t\right)\right\}_{i=1}^{L} \tag{95}$$

$$R^1_N = \left[\varphi\left(\frac{\Pi_t}{\Pi}-1\right)\frac{\Pi_t}{\Pi}\right] - (1-\epsilon) - \epsilon MC_t - \varphi\frac{\Pi^1_{t+1}}{R_t}\left[\left(\frac{\Pi^1_{t+1}}{\Pi}-1\right)\frac{\Pi^1_{t+1}}{\Pi}\frac{Y^1_{t+1}}{Y_t}\right] \tag{96}$$

$$R^1_M = D - \frac{1}{L}\sum_{i=1}^{L} B^i_t \tag{97}$$

$$R^1_{MN} = D - \frac{1}{L}\sum_{i=1}^{L} B^{i,1}_{t+1} \tag{98}$$

$$R^1_R = \tilde{Y}_t - C_t \tag{99}$$

$$R^1_{RN} = Y^1_{t+1} - C_{t+1} \tag{100}$$

where the function Ψ^{FB} is the Fischer-Burmeister, which can be used to represent

Kuhn-Tucker conditons. We will discuss the Fischer-Burmeister Condition below. Repeat the same steps as before, but now use the second random draw of the shocks. This allows to calculate the following objects

$$\left\{R_1^{i,2}\right\}_{i=1}^{L}, \left\{R_2^{i,2}\right\}_{i=1}^{L}, R_N^2, R_M^2, R_{MN}^2, R_R^2, R_{RN}^2 \tag{101}$$

3. Define the loss function:

$$R^2 = \sum_{i=1}^{L} \alpha_1^i R_1^{i,1} R_1^{i,2} + \sum_{i=1}^{L} \alpha_2^i R_2^{i,1} R_2^{i,2} + \alpha_N R_N^1 R_N^2 + \alpha_M R_M^1 R_M^2 + \tag{102}$$

$$\alpha_{MN} R_{MN}^1 R_{MN}^2 + \alpha_R R_R^1 R_R^2 + \alpha_{RN} R_{RN}^1 R_{RN}^2 \tag{103}$$

   where $\left\{\alpha_1^i\right\}_{i=1}^{L}, \left\{\alpha_2^i\right\}_{i=1}^{L}, \alpha_N, \alpha_M, \alpha_{MP}, \alpha_R, \alpha_{RP}$ determine the weights for the different equations

4. Optimize the parameters of the neural networks ψ_1 and ψ_2 to minimize the loss function with a stochastic gradient optimizer

5. Repeat steps 2 - 4for each batch (B times)

6. Simulate each batch economy b for $T^s im$ periods using randomly drawn shocks. This creates then the state vector for the next iteration of the optimizer

7. Repeat steps 2 - 6 N^{iter} times

Fischer-Burmeister Function The Fischer-Burmeister function can be used to capture computationally the complementary slackness conditions of the Karush-Kuhn-Tucker conditions. The complementary slackness conditions can be written for instance as:

$$e \geq 0, \quad f \geq 0, \quad e \times f = 0 \tag{104}$$

The Fischer-Burmeister function is defined as

$$\Psi^{FB}(e, f) = e + f - \sqrt{e^2 + f^2} \tag{105}$$

If $\Psi^{FB}(e, f) = 0$, then the complementary slackness conditions are satisfied.

We are interested using this to ensure that the borrowing constraint $B_t^i \geq \underline{B}$ is satisfied (see also equation (40)). In the algorithm, we used λ_t^i, which is defined as follows:

$$\lambda_t^i = 1 - \mu_t^i \tag{106}$$

The complementary slackness conditions can be written as

$$1 - \lambda_t^i \geq 0 \tag{107}$$

$$\left(B_t^i - \underline{B}\right) \geq 0 \tag{108}$$

$$\left(1 - \lambda_t^i\right) \times \left(B_t^i - \underline{B}\right) = 0 \tag{109}$$

and we minimize the Fischer-Burmeister condition to ensure that these conditions are hold

$$\Psi^{FB}\left(1 - \lambda_t^i, B_t^i - \underline{B}\right) \tag{110}$$

D Neural Network-Based Bayesian Estimation Algorithm

The following algorithm can be used to run a Neural Network-Based Bayesian Estimation

1. Train the model with the extended neural network approach to solve the model for the entire parameter space
2. Train a new neural network to save the result of the particle filter
3. Calculate the likelihood at chosen points (e.g. with Random Walk Metropolis Hastings Algorithm)

D.1 Neural Network-Based Particle Filter

1. Set up a neural network to approximate the likelihood function and intialize the network. The neural network Ω_{PF} maps the structural parameter values into a likelihood value:

$$L = \Omega_{PF}\left(\tilde{\Theta}|Data\right) \tag{111}$$

2. Use the particle filter to solve for the economy and calculate the likelihood value L^v(what we do at the moment)
3. Define the loss function

$$R^2 = (L - L^v)^2 \tag{112}$$

4. Optimize the parameters of the neural networks $\Omega_{PF}\left(\tilde{\Theta}|Data\right)$ to minimize the loss function
5. Repeat steps 2 to 4 B times

D.2 Estimation based on classical solution approach

We also use a Metropolis Hastings algorithm to estimate the parameters $\tilde{\Theta}$, when we solve the model with classical global solution methods. Our design of the algorithm follows Atkinson et al. (2020).[41]. We initially draw randomly from the prior distribution to get a proposal density for the Metropolis Hastings algorithm. We then run a burn-in period with the Metropolis

[41]See also Herbst and Schorfheide (2015) for useful

Hastings algorithm to get an updated proposal density. This proposal density is then used to start the final run of the Metropolis Hastings algorithm. The detailed approach is as follows:

1. 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 distirbution a candidate vector $\tilde{\Theta}^{New}$
   (b) Solve the model for the draw with a classical global solution method (e.g. with a time algorithm that uses piecewise linear policy function and approximates expectations with Gauss-Hermiture quadrature)
   (c) Use a particle filter to calculate the likelihood of the model $\ln L(\tilde{\Theta}^{New}|\text{Data})$ and combine it with the prior to obtain the log posterior $\ln g(\tilde{\Theta}^{New}|\text{Data})$
   (d) Repeat these steps N^{init} times and collect all draws as Θ^{init}
   (e) Approximate the covariance matrix with these draws
      i. Choose all draws where the likelihood is above the 90% quantile, which is denoted as $\hat{\Theta}$
      ii. Calculate the deviations of each draw from the mean: $\tilde{\Theta} = \hat{\Theta} - \overline{\hat{\Theta}}$
      iii. Calculate the covariance matrix: $\Sigma = (\tilde{\Theta}'\tilde{\Theta})/(0.1N^{init})$
      iv. Define the mode, that is the draw associated with highest likelihood, as $\tilde{\Theta}$
2. 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 $\tilde{\Theta}$ and covariance matrix $c\Sigma$, where the parameter is set to have an acceptance ratio between 20 and 40%.
   (a) Solve the model for the draw with a classical global solution method (e.g. with a time algorithm that uses piecewise linear policy function and approximates expectations with Gauss-Hermiture quadrature)
   (b) Use a particle filter to calculate the likelihood of the model $\ln L(\tilde{\Theta}^{New}|\text{Data})$
   (c) Combine the likelihood with the prior to evaluate the log posterior $\ln g(\tilde{\Theta}^{New}|\text{Data})$
   (d) Accept the draw if $\exp(\ln g(\tilde{\Theta}^{New}|\text{Data}) - \ln g(\tilde{\Theta}|\text{Data}))$ is larger than the draw from a standard uniform distribution. If the draw is accepted, the candidate density is updated to $\tilde{\Theta} = \tilde{\Theta}^{New}$
3. Repeat the previous step N^{Burn} times.
4. Use these N^{nurn} draws to get an updated candidate density. W
   (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: $\tilde{\Theta}^b = \hat{\Theta}^b - \overline{\hat{\Theta}^b}$
   (c) Calculate the covariance matrix: $\Sigma = (\tilde{\Theta}^{b\prime}\tilde{\Theta}^b)/(0.75N^{burn})$

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

5. Use the proposal density defined in the previous step and repeat step 2a N^{final} times
To be continued ...

D.2.1 Details on the Solution and Estimation Algorithm

To be continued ...

E Equilibrium Conditions

E.1 Linearized 3 equation NK model

$$\hat{X} = E_t\hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi\hat{\Pi}_t + \phi_Y\hat{X}_t - E_t\hat{\Pi}_{t+1} - \hat{R}^F_t\right) \tag{113}$$

$$\hat{\Pi}_t = \kappa\hat{X}_t + \beta E_t\hat{\Pi}_{t+1} \tag{114}$$

$$\hat{R}^F_t = \rho_A\hat{R}^F_{t-1} + \sigma(\rho_A - 1)\omega\sigma_A\epsilon^A_t \tag{115}$$

E.2 RANK model with ZLB

Add the equilibrium conditions of the RANK model

E.3 HANK model with ZLB

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:

$$\tilde{C}^i_t + \tilde{B}^i_t = W_t s^i_t \tilde{H}^i_t + \frac{R_{t-1}}{\Pi_t g_t}\tilde{B}^i_{t-1} - \tilde{T}^i_t + \tilde{Div}^i_t, \tag{116}$$

$$\tilde{\lambda}_t = \tilde{C}_t - h\frac{\tilde{C}_{t-1}}{g_t} \tag{117}$$

$$1 = \beta R_t \mathbb{E}_t\left[\left(\frac{\zeta_{t+1}}{\zeta_t}\right)\left(\frac{\tilde{\lambda}^i_t}{\tilde{\lambda}^i_{t+1}}\right)^\sigma \frac{1}{\Pi_{t+1}g^\sigma_{t+1}}\right] + \mu^i_t, \tag{118}$$

$$\chi(H^i_t)^\eta = (\tilde{\lambda}^i_t)^{-\sigma}\left(s^i_t\tilde{W}_t\right) \tag{119}$$

$$\tilde{Y}^j_t = N^j_t, \tag{120}$$

$$\tilde{W}_t = MC_t \tag{121}$$

$$\tilde{Div}_t = \tilde{Y}_t - W_t\tilde{Y}_t \tag{122}$$

$$\left[\varphi^R\left(\frac{\Pi_t}{\Pi} - 1\right)\frac{\Pi_t}{\Pi}\right] = (1-\epsilon) + \epsilon MC_t + \varphi^R\mathbb{E}_t\frac{\Pi_{t+1}}{R_t}\left[\left(\frac{\Pi_{t+1}}{\Pi} - 1\right)\frac{\Pi_{t+1}g^{\sigma-1}_t}{\Pi}\frac{\tilde{Y}_{t+1}}{\tilde{Y}_t}\right], \tag{123}$$

$$R^n_t = (R^N_{t-1})\rho_R\left(R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{\tilde{Y}_t}{\tilde{Y}}\right)^{\theta_Y}\right)^{1-\rho_R}, \tag{124}$$

$$R_t = \left[1, R_t^N\right] \tag{125}$$

$$\tilde{D}_t = \frac{R_{t-1}}{\Pi_t g_t}\tilde{D}_{t-1} - \tilde{T}_t \tag{126}$$

F Mapping the Model in the General Framework

F.1 Linearized 3 equation NK model

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

$$\mathbb{S}_t = \left\{\hat{R}_t^f\right\}, \qquad \text{and} \qquad \nu_t = \left\{\epsilon_t^A\right\}. \tag{127}$$

The control variables of the model are:

$$\psi_t = \left\{\hat{X}_t, \hat{\Pi}\right\}. \tag{128}$$

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

$$\bar{\Theta} = \{\}, \tag{129}$$

$$\tilde{\Theta} = \{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A\}. \tag{130}$$

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:

$$\begin{pmatrix}\hat{X}_t \\ \hat{\Pi}_t\end{pmatrix} = \psi_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right). \tag{131}$$

F.2 RANK model with ZLB

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

$$\mathbb{S}_t = \{\zeta_t\}, \qquad \text{and} \qquad \nu_t = \left\{\epsilon_t^\zeta\right\}. \tag{132}$$

The control variables of the model are:

$$\psi_t = \{C_t, N_t, T_t, Y_t, Div_t, MC_t\}. \tag{133}$$

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

$$\bar{\Theta} = \{\beta, \sigma, \eta, \epsilon, \chi, \Pi, Y\}, \tag{134}$$

$$\tilde{\Theta} = \{\theta_\Pi, \theta_Y, \varphi, \rho_\zeta, \sigma_\zeta\}. \tag{135}$$

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

$$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi_{NN}^{A}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right). \tag{136}$$

F.3 HANK model with ZLB

We recast the model to take out the stochastic trend in GDP growth, where we define variables as follows. $\tilde{X}_t = \frac{X_t}{Z_t}$. The detrended equilibrium conditions can be found in Appendix A.

We can map the HANK model in the general form of the outlined estimation procedure:

$$\mathbb{S}_t = \left\{\left\{\tilde{B}_{t-1}^i\right\}_{i=1}^{L}, \left\{s_t^i\right\}_{i=1}^{L}, R_{t-1}^N, \tilde{C}_{t-1}, \zeta_t, g_t, mp_t\right\} \tag{137}$$

$$\nu_t = \left\{\left\{\epsilon_t^{s,i}\right\}_{i=1}^{L}, \epsilon_t^{\zeta}, \epsilon_t^{g}, \epsilon_t^{mp}\right\} \tag{138}$$

As we approximate the distribution using 100 agents ($L = 100$), this which corresponds to 205 state variables S and 105 structural shocks.

The control variables of the model are

$$\psi_t = \left\{\left\{\tilde{C}_t^i\right\}_{i=1}^{L}, \left\{N_t^i\right\}_{i=1}^{L}, \left\{\tilde{B}_t\right\}_{i=1}^{L}, \left\{\mu_t\right\}_{i=1}^{L}, \tilde{T}_t, \tilde{Y}_t, \tilde{Div}_t, MC_t\right\} \tag{139}$$

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

$$\bar{\Theta} = \left\{\beta, \sigma, \chi, h, \varphi, \underline{B}, D, \epsilon, \varphi^R, g, \Pi, Y, \rho_r, \kappa_\Pi, \kappa_Y, \rho_s, \rho_z eta\right\} \tag{140}$$

$$\tilde{\Theta} = \{\sigma_s, \sigma_\zeta, \sigma_g, \sigma_{mp}\} \tag{141}$$

We use two neural networks to separate between individual and aggregate policy functions. The individual neural network solves for labor supply and the multiplier on the borrowing constraint:

$$\begin{pmatrix} \left\{N_t^i\right\}_{i=1}^{L} \\ \left\{\mu_t^i\right\}_{i=1}^{L} \end{pmatrix} = \left\{\psi_{NN}^{I}\left(\mathbb{S}_t^i, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}, \tag{142}$$

where $\mathbb{S}_t^i = \left\{\tilde{B}_{t-1}^i, s_t^i\right\}$. The neural network for the aggregate control variables determines the wage and inflation:

$$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi_{NN}^{A}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{143}$$

G Additional Results

G.1 Linearized 3 equation NK model

Figure 9 shows the policy function for inflation for variations of the parameters.

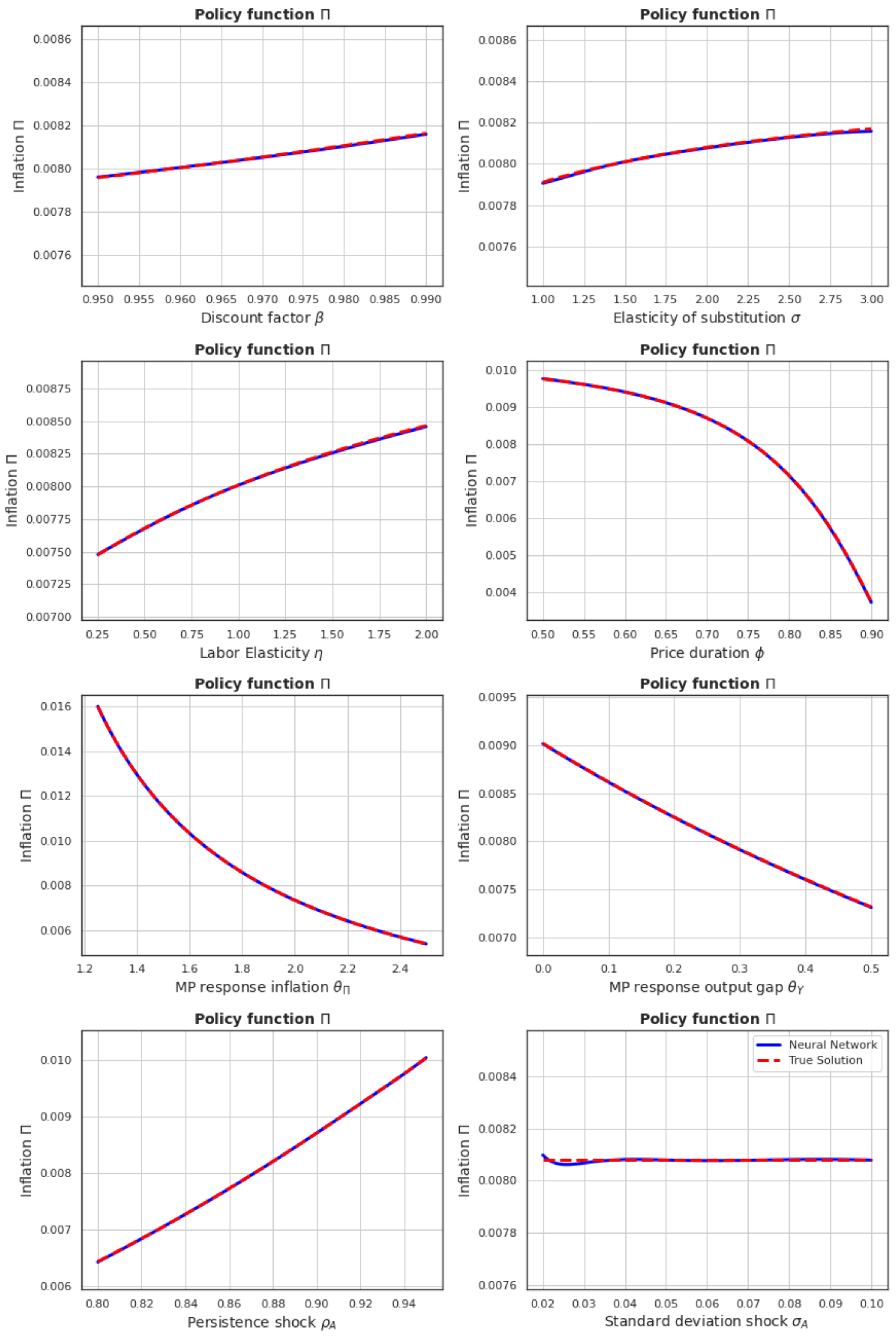

Figure 9: Comparison between the neural network based solution and the true analytical solution. The plot shows how variations in the structural parameter affect the policy function for inflation $\hat{\Pi}_t$. The policy function is evaluated at the one standard deviation of the ergodic distribution and the unvaried parameters are fixed at their mean.

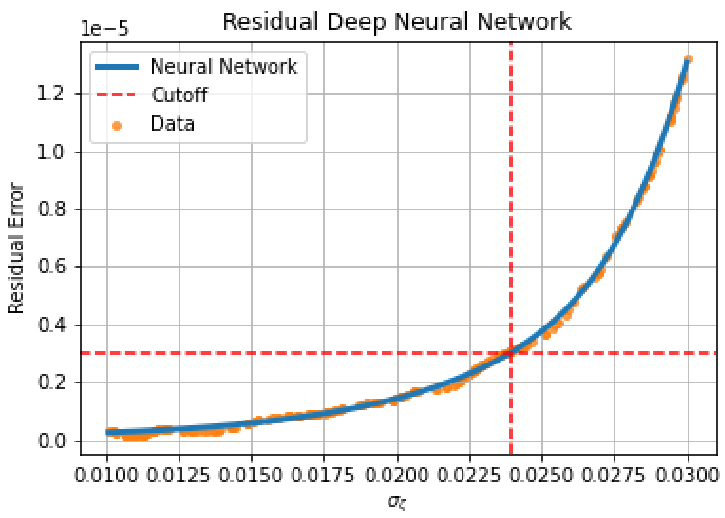

Figure 10: Surrogate neural network model that contains the residual error. It shows the trained neural network and contrasts it to the data that are used for training. The red line corresponds to a cut-off value at a residual error at .25e-5.

G.2 RANK model with ZLB

The surrogate neural network model that contains the residual error is shown in 10. 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. Figure 11 shows how the combination of parameters affect the residual error. The contour plot shows that the persistence and the standard deviation have the most impact on the residual error.

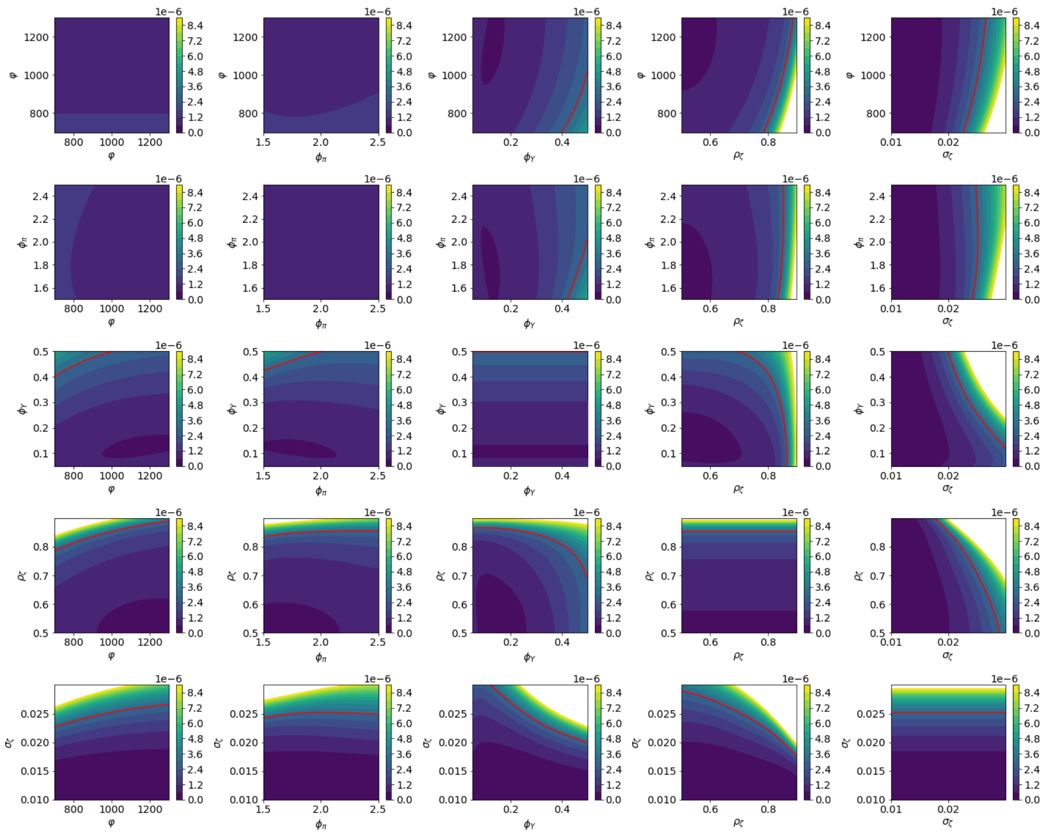

Figure 11: Surrogate neural network that contains the residual error is presented as contour plot. A dark value is associated with a low residual error, while a light value is associated with a high value. The red line is set at a cut-off value at 0.5e-5.