DEPARTMENT OF ECONOMICS

Essays in Quantitative Macroeconomics

Hanno Kase

Thesis submitted for assessment with a view to
obtaining the degree of Doctor of Economics
of the European University Institute

Florence, 21 December 2021

European University Institute
Department of Economics

Essays in Quantitative Macroeconomics

Hanno Kase

Thesis submitted for assessment with a view to
obtaining the degree of Doctor of Economics
of the European University Institute

Examining Board

Prof. David Levine, EUI (supervisor)
Prof. Jesus Bueren, EUI (co-supervisor)
Prof. Aldo Rustichini, University of Minnesota
Prof. Galo Nuño, Banco de España

Researcher declaration to accompany the submission of written work
Department Economics - Doctoral Programme

I Hanno Kase certify that I am the author of the work "Essays in Quantitative Macroeconomics" I have presented for examination for the Ph.D. at the European University Institute. I also certify that this is solely my own original work, other than where I have clearly indicated, in this declaration and in the thesis, that it is the work of others.

I warrant that I have obtained all the permissions required for using any material from other copyrighted publications.

I certify that this work complies with the Code of Ethics in Academic Research issued by the European University Institute (IUE 332/2/10 (CA 297).

I declare that this work consists of 26726 words.

I confirm that chapter 1 was jointly co-authored with Leonardo Melosi and Matthias Rottner and I contributed 33% of the work.

Signature and date:

Hanno Kase
14 December 2021

Abstract

This thesis consists of three essays in quantitative macroeconomics. In Chapter 1, joint with Leonardo Melosi and Matthias Rottner, we leverage recent developments in machine learning to develop methods to solve and estimate large and complex nonlinear macroeconomic models, e.g. HANK models. Our method relies on neural networks because of their appealing feature that even models with hundreds of state variables can be solved. While likelihood estimation requires the repeated solving of the model, something that is infeasible for highly complex models, we overcome this problem by exploiting the scalability of neural networks. Including the parameters of the model as quasi state variables in the neural network, we solve this extended neural network and apply it directly in the estimation. To show the potential of our approach, we estimate a quantitative HANK model that features nonlinearities on an individual (borrowing limit) and aggregate level (zero lower bound) using simulated data. The model also shows that there is an important economic interaction between the impact of the zero lower bound and the degree of household heterogeneity.

Chapter 2 studies the impact of macroprudential limits on mortgage lending in a heterogeneous agent life-cycle model with incomplete markets, long-term mortgage, and default. The model is calibrated to German economy using Household Finance and Consumption Survey data. I consider the effects of four policy instruments: loan-to-value limit, debt-to-income limit, payment-to-income limit, and maximum maturity. I find that their effect on homeownership rate is fairly modest. Only the loan-to-value limit significantly reduces the homeownership rate among young households. At the same time, it has the largest positive welfare effect.

Chapter 3 explores applications of the backpropagation algorithm on heterogeneous agent models. In addition, I clarify the connection between deep learning and dynamic structural models by showing how a standard value function iteration algorithm can be viewed as a recurrent convolutional neural network. As a result, many advances in the field of machine learning can carry over to economics. This in turn makes the solution and estimation of more complex models feasible.

To my family and Fatima

Acknowledgements

I have benefited greatly from many smart and kind people during the writing of this thesis.

First, I would like to express my gratitude to my supervisors, David Levine and Jesús Bueren, for their advice and support during my PhD. I would not be the economist I am today without their guidance as well as the freedom they gave me to wander and explore.

I am grateful to the external committee members, Aldo Rustichini and Galo Nuño, for kindly agreeing to be part of the thesis committee. Galo's pioneering research in using neural networks as part of the algorithm for solving macro models was what got me interested in deep learning in the first place. Working together with Aldo during the peak of the pandemic on a high dimensional dynamic programming problem led me to re-evaluate the direction of my research towards techniques that help solve more complex models. I am thankful for the opportunity to continue in this direction together with him during my post-doc and beyond.

I would like to thank my coauthors of the first chapter, Leonardo Melosi and Matthias Rottner. It has been, and will undoubtedly continue to be, an incredibly productive and fun collaboration.

Five years in Florence would not have been nearly as enjoyable without the wonderful and brilliant classmates and friends. I am extremely lucky to have met my partner, Nurfatima Jandarova, the kindest and smartest person in my eyes. I am always happy to help her and she has been an immeasurable help and source of encouragement for me, always.

I am forever grateful to my parents, Anne and Enno, and my brother and sister, Heiko and Anna-Helena, for their encouragement and unconditional support throughout my life.

I would also like to thank Lenno Uusküla, Karsten Stæhr, Tõnn Talpsepp and Peeter Luikmel: their encouragement led me to pursue a PhD at the EUI.

Finally, I would like to thank Education and Youth Board of Estonia for funding me during the PhD program and the Economics and Research Department of the Bank of Estonia for the opportunity to participate in their visiting researcher program.

Table of contents

Chapter 1

Solving and Estimating Macroeconomic Models of the Future

joint with Leonardo Melosi and Matthias Rottner

1.1 Introduction

In recent years, heterogeneous agents models have been gaining more and more prominence in the economic literature to study matters like the importance of borrowing limits, social inequality, and the transmission of monetary policy. However, these models are typically hard to solve because of their elevated complexity, forcing economists to study tractable approximations of these models and limiting their empirical analysis. This tractability often comes at the cost of losing interesting features of the model, such as the impact of the zero lower bound, stochastic volatility or other aggregate nonlinearities. Our paper proposes an approach based on machine learning to solve highly complex models.

We develop a novel neural network based likelihood estimation procedure that exploits the advantages of neural networks. Neural networks, which are an efficient method that is used by data scientist to learn fast about highly complicated mathematical functions or mappings, can capture the highly complex solution mapping of nonlinear models. However, classical estimation method requires the repeated solving of the model for different parameters. Even though neural networks can handle such complicated models, it is computationally infeasible to solve such models sufficiently often. We overcome this problem by exploiting the scalability of neural networks. By including the parameters of the model as quasi state

variables in the neural network, we solve this extended neural network and can then directly condition our solution on the parameters. An additional challenge for the estimation is that the fit of the model with the data (the likelihood) also requires a repeated evaluation with a particle filter. Therefore, we propose an approach that involves training an additional neural network to learn the outcome of the particle filter. Using these two elements, we can now estimate models that have been out of reach. As an example, we use our method to solve and estimate a quantitative HANK model that features nonlinearities on an individual and aggregate level using simulated data.

Our approach relies on neural networks as these can handle large amounts of inputs and are thus predestined for this task. Specifically, neural networks are very scalable, which implies that additional extra inputs can be added at low costs regarding computational power and time. This enables neural networks to solve large and complex models. In other words, neural networks can 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 classical global solution methods face. Although the complexity of the neural network problem comes at lower costs, a single solution of such elaborated models is still very time-consuming. This is a particularly challenging issue because (Bayesian) estimation usually requires that a model is solved hundreds of thousands times to evaluate different parameter combinations. This problem seems to render a classical estimation strategy impossible.

To overcome this critical problem, we redefine the estimation strategy such that it requires a single run. The key feature of this trick is that neural networks can handle large amounts of input. We treat the parameters that we want to estimate as direct inputs to the neural network. The parameters are now considered as quasi-state variables. This results in an extended neural network that has state variables and parameters simultaneously as inputs. We then adapt a neural network based solution so that it can handle such an extended neural network. We now need to find only once the solution for this extended neural network to proceed with the estimation. The problem becomes now computationally and time-wise feasible[1]. As a consequence, our approach allows the estimation of macroeconomic models with many state variables.

A second bottleneck is that the fit of the model with the data needs to be evaluated at each for the different parameter combinations. As the solution of the model is nonlinear, we cannot use the Kalman filter. Instead, the particle filter is applied to obtain the likelihood

1. It is more challenging to find the solution of the extended neural network due to its increased complexity. However, the additional costs are negligible relative to repeatedly solving the original model.

function. The execution of a particle filter for a model with hundreds of state variables is very costly. This makes it impossible to generate the number of draws usually generated during an estimation procedure. To get around this issue, we propose a new particle filter method based on neural networks. Instead of using the particle filter for each draw, we train a new additional neural network to approximate the results of the particle filter. The particle filter calculates the likelihood at randomly drawn parameter combinations based on the extended neural networks. The calculated likelihoods then serve as data points to train a neural network that can approximate the outcome of the particle filter. As a consequence, we can evaluate the likelihood of large models with hundreds of state variables sufficiently fast.

We use our approach to solve and estimate a nonlinear HANK model that contains both idiosyncratic and aggregate risk using simulated data. 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 zero lower bound for the monetary authority. Taken all together, the model features hundreds of state variables: aggregate shocks, aggregate states, an idiosyncratic shock and asset holdings for each agent. To provide a controlled environment for the estimation of the nonlinear HANK model, we use a realistically calibrated model to generate simulated data and assess if the estimation can recover the true parameters. Before moving on to the estimation, we assess the precision of our solution. Specifically, we compare the outcome of the machine learning technique with classical solution methods. As the outlined model is too complex to be solved with classical solution techniques, we use the nonlinear representative agent version for comparison. This shows that the outcomes of the different solution methods fit together.

To assess the potential of our neural-network-based Bayesian estimation method, we estimate the standard deviation of the idiosyncratic and aggregate shocks of this model. Our observables are three aggregate time series. We use simulated data for output growth, inflation and the nominal interest rate, which have been used in the literature with representative agents to analyse the spell of the zero lower bound episode in the US. Using the extended neural network and the neural-network-based particle filter, we obtain the posterior based on one million draws. We find that the estimated posterior mode is close to the true value. This shows that we can estimate the described nonlinear HANK model. Importantly, the estimation is completed in less than a day using a modern desktop computer[2].

Finally, we establish the importance of nonlinearities for the solution. We find that heterogeneity and the zero lower bound interact. It affects the time-series dynamics and

2. We use an AMD Ryzen 9 3900X 12-Core processor (CPU) and Nvidia GeForce GTX1080ti (GPU) for neural-network-based Bayesian estimation.

the distribution of asset holdings. This again emphasizes the importance of methods that can tackle the estimation of globally solved HANK models. More generally, the developed methods can be used to assess the connection between heterogeneity and aggregate risk based on empirical macroeconomic models.

To sum up, the paper contains three main contributions. The first contribution is to propose a new neural-networks-based Bayesian estimation procedure. The second contribution is to use a particle filter that is based on neural networks. Finally, we estimate a nonlinear HANK model with simulated data in a laboratory setup.

Literature. The paper is connected to the literature that develops global solution methods using neural networks as the fundamental building block in the solution of complex dynamic economics models. In particular, we rely on the approach of Maliar, Maliar, and Winant (2021) that can handle models with hundreds or even thousands of state variables as demonstrated in a setup with heterogeneous agents. The solution approach is designed for discrete-time models and uses an innovative idea to approximate expectations. Adapting their solution approach to directly include parameters as inputs, we introduce a novel Bayesian estimation method. The comparative advantage of this new approach is that it can estimate large macroeconomic models. Other exciting approaches with neural networks are developed in Azinovic, Gaegauf, and Scheidegger (2019), Duarte (2018b), Fernández-Villaverde et al. (2020), and Villa and Valaitis (2019), among others. These approaches could be also combined with our Bayesian estimation strategy.

Norets (2012) explores the inclusion of parameters in an estimation of a dynamic discrete choice models using neural networks. He also provides theoretical justifications for this method. Duarte (2018a) also follows this approach for a method-of-moments estimation. Three major conceptual differences in our work are that our developed method focuses on Bayesian estimation, is designed for large and complex models and can handle a setup with heterogeneous agents. We also propose a new approach to evaluate the fit with the data, which requires a nonlinear filter. An additional neural network is trained to learn the outcome of the particle filter.[3] This step allows to execute the particle filter considerably less often.

Finally, our paper is related to the HANK literature. There are only very few papers that solve a nonlinear HANK model globally. Fernández-Villaverde et al. (2021) study the connection between the zero lower bound and inequality in nonlinear HANK model. Rel-

3. Chen, Didisheim, and Scheidegger (2021) develop an estimation approach, where they train a deep neural network, a so-called surrogate model, on a dataset generated from the full model with different parameter values.

ative to their work, we use a more complex model that features several aggregate shocks.[4] Maliar and Maliar (2020) also solve a HANK model globally with neural networks. Furthermore, Gorodnichenko et al. (2021) assess the impact of idiosyncratic stochastic volatility on consumption and portfolio choices. While these three mentioned papers rely on neural networks, Schaab (2020) also studies a nonlinear HANK model based on a solution method unrelated to machine learning. We contribute to the literature by outlining and demonstrating with simulated data an estimation method that captures the idiosyncratic and aggregate risk of quantitative nonlinear HANK models. Estimations of HANK models have so far been restricted to an approximation of aggregate risk to a first order. Bayer, Born, and Luetticke (2020) estimates a a quantitative model with US data, while Lee (2020) extends this approach to account partially for occasionally binding constrains in line with the idea of Guerrieri and Iacoviello (2015). Auclert et al. (2019) provide another estimation method that uses perfect foresight for aggregate shocks.

Outline. The paper is organized as follows. In Section 1.2, we provide a general overview of neural networks and highlight two remarkable features of neural networks that we exploit for a neural-networks-based Bayesian estimation. In Section 1.3, we discuss how neural networks can be used to solve a very large class of macroeconomic models that can also include heterogeneity. In Section 1.4, we outline the neural-networks-based Bayesian estimation method, where we treat parameters as quasi-state variables. We also present how we can facilitate the execution of the particle filter with neural networks. In Section 1.5, we solve and estimate a quantitative nonlinear HANK model with our developed approach. Additionally, we emphasize the importance of capturing nonlinearities on an individual and aggregate level simultaneously. Section 1.7 concludes the paper.

1.2 Deep Learning and Neural Networks

Deep learning is a class of machine learning techniques with (deep) neural networks as the fundamental building block[5]. In this paper, we outline a neural-networks-based solution and estimation approach that can be applied to complex and large macroeconomic models

4. Fernández-Villaverde et al. (2021) use a hybrid method that combines classical solution method with a neural network approach for the aggregate law of motion. The advantage of their approach is that they do not need to approximate continuum of agents. However, the classical solution method entails the curse-of-dimensionality so that the size of the model is limited. Therefore, our approach can handle a more complex model with multiple aggregate shocks and aggregate state variables.

5. Goodfellow, Bengio, and Courville (2016) is a very good overview for artificial intelligence and in particular deep learning. Fernández-Villaverde et al. (2020) and Maliar, Maliar, and Winant (2021) also discuss machine learning in the context of macroeconomic modelling.

previously considered out of reach. To achieve this task, we utilize two remarkable features of neural networks. The first feature is that the neural networks can approximate any function as long as the network is sufficiently large. This is the so-called universal approximation theorem. The theorem implies that (large) macroeconomic models may, in principle, be approximated with neural networks. The second feature is that the neural network can handle a large number of inputs. Specifically, a neural-networks-based solution method is very scalable: additional inputs can be added at low computational costs. As a consequence, neural networks can tackle the curse of dimensionality and can in practice capture large models. In addition to solving complex models, the universal approximation theorem and the scalability of neural networks aid in making the estimation of such models feasible. We thereby provide a novel path to the estimation of macroeconomic models.

In this section, we present a short primer on the design and training of neural networks. Afterwards, we discuss the universal approximation theorem and the scalability in more detail.

1.2.1 Deep Neural Networks

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

$$Y = \psi_{NN}(S|W) \tag{1.1}$$

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 into 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, which is again visible. 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 number of layers determines the depth of the neural network. A neural network with more than one hidden layer is classified as deep. Figure 1.1 provides a schematic example of a deep neural network with three inputs, two hidden layers with six neurons each, and four outputs. The displayed layers are said to be dense as they are fully connected to each input from the previous layer.

The neural network composes mathematical functions that are performed at each neuron in the network. The value of a single neuron is calculated as the weighted sum of its inputs $s_1, s_2, \ldots, s_S$ with weights $w_1, w_2, \ldots, w_S$ and is adjusted by a bias/constant w_0[6]. The adjusted

6. Depending on the position of the neuron in the network, the inputs are given by the value of neurons either in the input layer or in the previous layer.

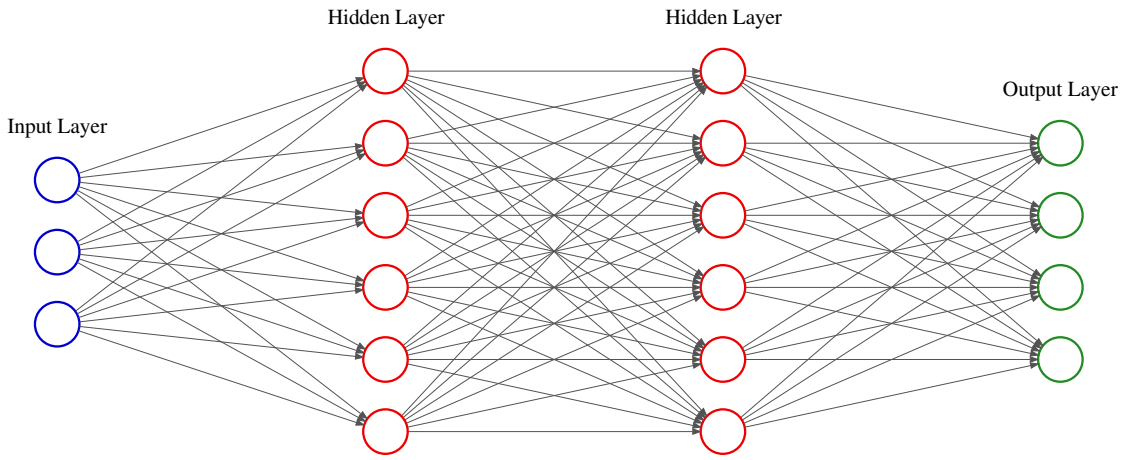


Figure 1.1: Example of a deep neural network.

weighted sum is then taken to a nonlinear activation function $h(\cdot)$, which could be for instance a ReLU (rectified linear unit) or hyperbolic tangent function. The result is a single output $\tilde{y}$:

$$\tilde{y} = h(w_0 + \sum_{i=1}^{S} w_i s_i) \tag{1.2}$$

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 the Figure 1.1, these neurons are stacked to form the neural network. The entire neural network can then be summarized using the parameter vector W with weights at each neuron. In the next paragraph, we discuss how we train these parameters.

Loss Function and Training. In this paragraph, we discuss how we 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 the mean squared error loss Φ^L:

$$\Phi^L(W) = 1/B \sum_{i=1}^{B} (Y_i - \psi_{NN}(S_i|W))^2 \tag{1.3}$$

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.[7] This loss is evaluated using a total batch B of data points.

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

$$W = \arg\min_{W} \Phi^L(W) \tag{1.4}$$

The optimization relies on an 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. It allows us to compute how a change in weights affects the final loss. The step size in updating the weights depends on the learning rate. The learning rate should be set sufficiently high to avoid local minima and sufficiently low to ensure convergence.

One problem with neural networks is overfitting. To avoid this problem, the data is separated usually in a training and test subsets. 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 neural network is usually done on graphics processing units (GPUs) as these can be used to parallelize many but rather simple operations. PyTorch and TensorFlow are popular open source machine learning libraries that can be used to build and train neural networks.

1.2.2 Universal Approximation Theorem

An important argument for the use of neural networks is the universal approximation theorem. It states that a sufficiently wide feedforward[8] network with at least one hidden layer can approximate any function in a finite-dimensional space with any desired non-zero error given a sufficient width (Hornik, Stinchcombe, and White 1989; Cybenko 1989; Bach 2017).

7. 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.

8. That is, a network without any loops or cycles such that information only moves forward.

Importantly, macroeconomic models can be cast in such a finite-dimensional space. As a consequence, neural networks can, in theory, be applied to solve macroeconomic models.

1.2.3 Scalability

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

However, neural networks can alleviate 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 the problem, while in the case of traditional functional approximators the size of the problem grows exponentially (Fernández-Villaverde et al. 2020). This scalability of neural networks allows them to handle models with a large number of states and alleviate the curse of dimensionality. As a consequence, neural networks can also, in practice, be applied to solving complex macroeconomic models.

But, there is more to it than this. We explore how to use the scalability of neural networks to estimate macroeconomic models. The key insight 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 defined by a different combination of parameter values. Importantly, the universal approximation theorem also provides the theoretical groundwork for this approach as we only need a neural network with sufficient width to include these additional inputs. Exploiting the two features, we are able to outline a general neural-networks-based solution and estimation method.

1.3 Neural Network-Based Solution Method

Deep learning can be used to solve complex nonlinear macroeconomic models that have previously been considered out of reach in terms of computational complexity. The outlined approach uses deep neural networks to solve models and is based on Maliar, Maliar, and Winant (2021). The advantage of this deep neural network approach is that the solution method is scalable, which allows it to handle high-dimensional models. As a consequence, it can significantly ease the curse of dimensionality that more classical global solution method face. The scalability is the reason that the approach can be applied to models featuring rich

heterogeneity, many states, several structural shocks or all these elements combined as we will demonstrate with an example later in this paper. In fact, the scalability of the approach allows not only to solve such models, but also estimate them as we will demonstrate later.

1.3.1 Nonlinear Model Representation

We are interested in solving a dynamic stochastic general equilibrium model 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:

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t | \Theta\right), \tag{1.5}$$

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 $\psi(\cdot)$:

$$\psi_t = \psi(\mathbb{S}_t|\bar{\Theta}), \tag{1.6}$$

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.[9] For this reason, the focus is only on the policy functions $\psi(\mathbb{S}_t|\bar{\Theta})$.

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

$$F(\psi(\mathbb{S}_t|\bar{\Theta})) = 0, \tag{1.7}$$

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

where the function space is derived in line with Euler equation methods. To solve this, we use a deep learning approach with neural networks to approximate the policy functions of the macroeconomic model.

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), sectors or banks. Heterogeneity usually assumes the existence of a continuum of agents, which implies that the distributions of individual states and shocks are infinite dimensional:

$$\int \mathbb{S}_t^i d\Omega \quad \text{and} \quad \int \nu_t^i d\Omega, \tag{1.8}$$

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, Maliar, and Winant (2021).[10] This allows to capture the continuous distribution with a large but finite number of states.[11] 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:

$$\left\{\mathbb{S}_t^i\right\}_{i=1}^L \quad \text{and} \quad \left\{\nu_t^i\right\}_{i=1}^L. \tag{1.9}$$

The state variables and shock can be summarized as:

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} \quad \text{and} \quad \nu_t = \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}, \tag{1.10}$$

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

In a similar fashion, we adjust the policy functions:

$$\psi_t^i = \psi^I(\mathbb{S}_t^i, \mathbb{S}_t|\bar{\Theta}) \quad \text{and} \quad \psi_t^A = \psi^A(\mathbb{S}_t|\bar{\Theta}). \tag{1.11}$$

10. Agents assume in their maximization problem that their individual weight is zero.

11. 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.

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 when additionally conditioned on the individual variables.

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

$$\mathbb{S}_t = \left\{ \{\mathbb{S}_t^i\}_{i=1}^L, \mathbb{S}_t^A \right\} = f\left(\left\{ \{\mathbb{S}_{t-1}^i\}_{i=1}^L, \mathbb{S}_{t-1}^A \right\}, \left\{ \{\nu_t^i\}_{i=1}^L, \nu_t^A \right\} | \Theta \right) \tag{1.12}$$

$$\psi_t = \left\{ \{\psi_t^i\}_{i=1}^L, \psi_t^A \right\} = \left\{ \{\psi^I(\mathbb{S}_t^i, \mathbb{S}_t | \bar{\Theta})\}_{i=1}^L, \psi^A(\mathbb{S}_t | \bar{\Theta}) \right\} \tag{1.13}$$

The number of individual and aggregate state variables are S^i and S^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.

1.3.2 Solving the Model with Neural Networks

We use a neural network based solution method to find the policy functions. We train the neural networks to map the S inputs (state variables) into a number of output variables O (policy functions), which gives a numerical solution for the macroeconomic model. Before providing more details on the approach, an example of a policy function that is trained with neural networks is given in Figure 1.2. The policy function can capture nonlinear dynamics, which is essential for an economic model with idiosyncratic and aggregate nonlinearities.

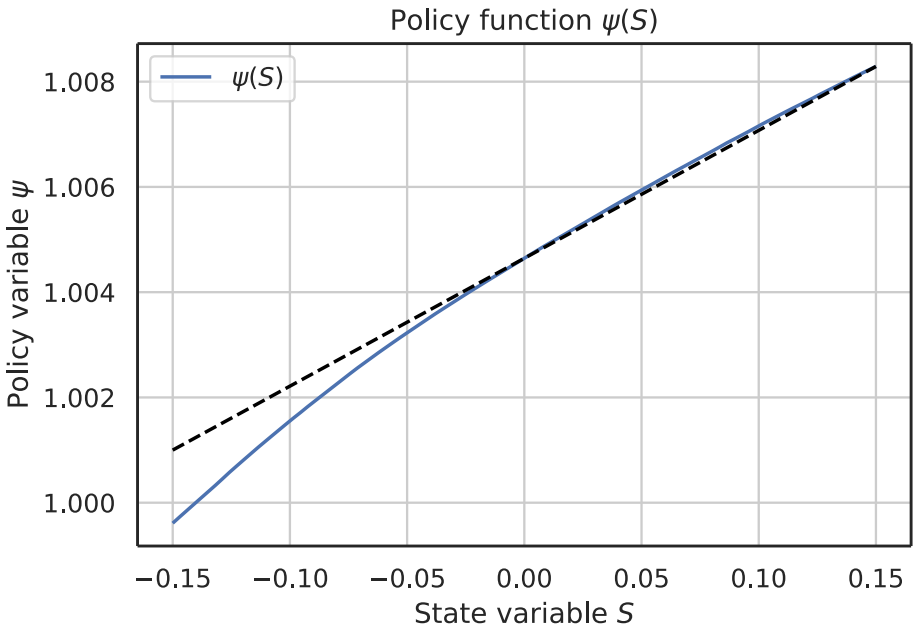


Figure 1.2: Example of a policy function obtained with neural networks solution approach.

Neural Networks and Policy Functions We use neural networks to approximate the individual and aggregate policy functions, ψ^I and ψ^A. In particular, we set up two neural networks, ψ^I_{NN} and ψ^A_{NN}, that approximate the individual and aggregate policy, respectively. The mapping from the state variables to the control variables with neural networks as function approximator is:[12]

$$\psi^i_t = \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t; \Theta\right), \tag{1.14}$$

$$\psi^A_t = \psi^A_{NN}\left(\mathbb{S}_t; \Theta\right). \tag{1.15}$$

The entire vector of control variables is given as:

$$\psi_t = \left\{ \left\{ \psi^i_t \right\}_{i=1}^L, \psi^A_t \right\} = \left\{ \left\{ \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t; \Theta\right) \right\}_{i=1}^L, \psi^A_{NN}\left(\mathbb{S}_t; \Theta\right) \right\} \tag{1.16}$$

For simplicity, we define $\psi_{NN}\left(\mathbb{S}_t; \Theta\right) \equiv \left\{ \left\{ \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t; \Theta\right) \right\}_{i=1}^L, \psi^A_{NN}\left(\mathbb{S}_t; \Theta\right) \right\}$. This formulation highlights that approach solves simultaneously for the individual law of motion and the aggregate law of motion.[13]

As mentioned, once we know these policy functions, it is straightforward to solve for the transition equation and recover the law of motion for the state variables. We now discuss is how we fit the neural networks.

Loss Function and Training of the Neural Networks The next step is to train the neural network to approximate the policy functions. For this, the neural networks ψ^I_{NN} and ψ^A_{NN} are trained to minimize a defined loss function. We suggest to solve the set of first-order conditions, which can be used to characterize the solution:

$$E_t\left[\Gamma(\mathbb{S}_{t+1}, \psi_{t+1}, \nu_{t+1}, \mathbb{S}_t, \psi_t, \nu_t)\right] = \mathbf{0}. \tag{1.17}$$

where Γ is a nonlinear first-order system of difference equations.

It should be noted that in a setup with heterogeneity, this set of equations contains the individual and aggregate equilibrium conditions. The number of equations would then correspond to the number of policy functions $O = O^i \times L + O^A$. However, another advantage of the neural network approach is that it does not prevent us from adding extra equations.

12. 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. However, our exposition can be generalized easily to other scenarios with only one or more neural networks.

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

We define the loss function as the weighted mean of the squares of the errors:

$$\phi^L = \frac{1}{K}\sum_{k=1}^{K} \alpha_k (E_t\left[\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1})\right])^2, \tag{1.18}$$

where α_k determines the weight of each equation.[14]

The neural networks ψ_{NN} are trained with a batch B. This can be thought of as having B economies in parallel. This allows the neural network to be trained simultaneously for a large variety of state values and shocks, which improves the precision of the solution. The loss function with a batch size B can be written as:

$$\Phi^L = \frac{1}{B}\sum^{B}\frac{1}{K}\sum_{k=1}^{K} \alpha_k (E_t\left[\Gamma_k^K(s_{t+1,b}, \psi_{t+1,b}, s_{t,b}, \psi_{t,b}, \nu_{t+1,b})\right])^2 \tag{1.19}$$

A stochastic gradient descent method, which minimizes the loss functions, updates iteratively the parameters of the neural network on a stochastic solution space. As a consequence, we minimize the errors of $B \times K \geq B \times (O^i \times L + O^A)$ equations in each iteration.

Stochastic State Space and Simulations The neural network is trained on stochastic solution domain. We are interested in training the algorithm only on the relevant n-dimensional domain of the state space. We ensure this via simulation step in our solution algorithm that approximates the ergodic distribution of the model. After maximizing the neural network for the given draw 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.

Expectations A particular problem in such a setup with potential hundred of shocks is to evaluate expectations, which requires to evaluate high-dimensional integrals. To overcome this critical issue, we approximate the expectations with the all-in-one expectation method of Maliar, Maliar, and Winant (2021). This method approximates the expectations of next period values using two uncorrelated independent draws for the next period shocks, that is

14. As 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.

ν_{t+1}^1 and ν_{t+1}^2 realizations. Therefore, we can approximate the expectations as follows:

$$E_t \left[\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1})\right])^2 \tag{1.20}$$

$$\approx E_{\nu^1}\left[\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1}^1)\right] E_{\nu^2}\left[\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1}^2)\right] \tag{1.21}$$

$$= E_{\nu^1\nu^2}\left[\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1}^1)\Gamma_k^K(s_{t+1}, \psi_{t+1}, s_t, \psi_t, \nu_{t+1}^2)\right] \tag{1.22}$$

where we use for the last row that the draws are independent.[15]

The loss function can then be written as:

$$\Phi^L = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{K}\sum_{k=1}^{K}\alpha_k\left(E_{\nu^1\nu^2}\left[\Gamma_k^K(s_{t+1,b}, \psi_{t+1,b}, s_{t,b}, \psi_t, \nu_{t+1,b}^1)\Gamma_k^K(s_{t+1,b}, \psi_{t+1,b}, s_{t,b}, \psi_{t,b}, \nu_{t+1,b}^2)\right]\right) \tag{1.23}$$

This method allow to evaluate expectations in a efficient way and can be used in such a stochastic setup that allows to draw the shocks randomly. The efficiency of this approximation method is very important as this allows to solve and estimate models with more than 100 structural shocks as we will show in our example.

Algorithm A detailed description of the algorithm to solve macroeconomic models is in the Appendix 1.A. Our codes rely on *PyTorch* as the machine learning framework and Adam (Kingma and Ba 2014) as the stochastic optimizer.

1.4 Neural Network-Based Bayesian Estimation

The crucial contribution in this paper is to go beyond solving the complex models to estimating them using neural networks. We develop a new neural-networks-based (Bayesian) estimation technique that accomplishes this challenging task.[16] The outlined approach opens up the possibility to estimate models that would have seen out of reach due to their complexity until this point. The approach can be used to solve and estimate a HANK model in its fully nonlinear specification that captures idiosyncratic and aggregate risk as we demonstrate in this paper. However, it can also be applied to other economic models such as a large nonlinear representative agent model or a multi-country setup.

15. This is based on a result from probability theory. Two random variables α and β, which are independent and from the same distribution, have that $E[\alpha]^2 = E[\alpha]E[\beta] = E[\alpha\beta]$ (Maliar, Maliar, and Winant 2021).

16. While we focus on Bayesian estimation, our outlined elements could be also used for other estimation procedures such as method of moments.

Before outlining our approach, it is instructive to describe the usual steps of a Bayesian estimation procedure. The first step is to draw a candidate vector for the parameters that are estimated from a proposal density. Second, the model is solved specifically for the drawn parameters. Third, a filter (e.g. particle filter as the setup is nonlinear) evaluates the likelihood of the solved model. Fourth, the draw is accepted or rejected and the proposal density adjusted. Usually, these steps are repeated hundreds of thousands of times to estimate the model. However, this strategy exhibits two important bottlenecks. First, despite its scalability, repeatedly solving a model with neural networks is very time-consuming. Such approach would render estimation infeasible. Second, the execution of a nonlinear filter also slows down the estimation and makes it very costly to generate the number of draws usually executed in such an estimation procedure.

To overcome the critical issue of repeatedly solving the model, we outline a new approach that exploits the scalability of the neural-networks-based solution method. The crucial gambit is that we solve the model only once, but include the entire parameter space that we want to estimate as input. For this, we extend the state space of the neural network and treat the parameters as quasi-state variables in the spirit of Norets (2012)[17]. While it can take some time to solve a model with hundreds of state variables, control variables and structural shocks using neural networks, adding the parameters as quasi-state variables has comparatively little impact on the computation time, especially compared to solving the model repeatedly.[18]. Therefore, the exploitation of the scalability of neural networks creates this new possibility of estimating complex nonlinear models.

The second problem is related to the costly computations of the likelihood with the particle filter. We suggest a novel neural-networks-based particle filter method to get around this issue. Instead of repeatedly applying the particle filter to each draw, we train a new additional neural network to approximate the results of the particle filter. To do this, we calculate the likelihoods for different parameter combinations from the solved parameter space with a particle filter. Importantly, the calculation of the likelihoods can be parallelized, which can be used to reduce another bottleneck in an estimation procedure. Treating the calculated likelihoods as data, we can then use these data points to train the neural network. To avoid overfitting of the neural network, we split the calculated points into a training and testing

17. Norets (2012) estimates a dynamic discrete choice model. Duarte (2018a) uses a similar approach for a method-of-moments approach.

18. In practice it can also be helpful to increase the size of the neural networks to capture additional complexity from the quasi-state variables.

sample.[19] This neural network based filter approach makes it possible to filter models with hundreds of state variables sufficiently fast.

As a consequence, we have already solved the model and obtained the likelihood before we start drawing from the proposal density. Once we start to actually draw from a proposal distribution, it takes us less than one millisecond to evaluate a single draw from the proposal density, which enables an estimation of large and complex models with millions of draws. In that regard, we can have similar or even longer chains as with a Metropolis Hastings algorithm that is normally used in a linear setup.

Extended Neural Network. The crucial key is that we solve the model only once, but we account in this step for the entire set of parameters that we want to estimate. The set of parameters can be divided in two subsets

$$\Theta = \{\tilde{\Theta}, \bar{\Theta}\}, \tag{1.24}$$

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 when we solve the economy. The extended neural network can then be written as

$$\psi_t^i = \psi_{NN}^I \left(\mathbb{S}_t^i, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right), \tag{1.25}$$

$$\psi_t^A = \psi_{NN}^A \left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{1.26}$$

where now the values for the parameters $\tilde{\Theta}$ are treated as state variables.

The total number of inputs in the extended neural network is then the number of parameters to be estimated P (pseudo state variables) and the true state variables ($S_t^i \times L + S_t^A$). This strategy is possible as we exploit the scalability of the neural network in handling many inputs.

We simultaneously solve the economy for the entire parameter space:

$$\tilde{\Theta} = \left\{ \left[\tilde{\Theta}^{\underline{1}}, \tilde{\Theta}^{\bar{1}}\right], \left[\tilde{\Theta}^{\underline{2}}, \tilde{\Theta}^{\bar{2}}\right], \ldots, \left[\tilde{\Theta}^{\underline{P}}, \tilde{\Theta}^{\bar{P}}\right] \right\}, \tag{1.27}$$

where $\tilde{\Theta}^{\underline{i}}$ and $\tilde{\Theta}^{\bar{i}}$ is the lower bound and upper bound, respectively, for the pseudo parameter i that will be estimated. We solve the neural network for a bounded space to increase precision. However, the bounds are conceptually not necessary and we could draw from a random distribution for the parameters. Another advantage of this approach is that we always solve for

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

the entire equilibrium. Therefore, we can easily estimate parameters that affect the stochastic steady state. In other words, the extended neural-network does not impose any restrictions, which parameters can be treated as quasi state variables. When solving the model, we randomly draw from the parameter space.[20] This allows to train the network simultaneously for the entire parameter space. Otherwise, the steps in training the extended neural networks are the same as in Section 1.3, in which the neural network featured only state variables.

The decisive difference is now that the neural network approximates the policy functions for an entire set of different structural parameters.[21] As a consequence, we need to solve the model only once!

A graphical characterization of the extended neural network and its connection to the policy functions can be seen in Figure 1.3. The left panel shows the policy functions for different parameter values. This shows that the mapping from state variables to policy variable depends on the chosen model parameter. Importantly, these different figures are created with the same neural network. The right panel further illustrates the idea of treating the model parameters as quasi 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 directly connected. The points in the left plot at $S = 0.00$ correspond to the policy variable on the right axis for the chosen parameter.

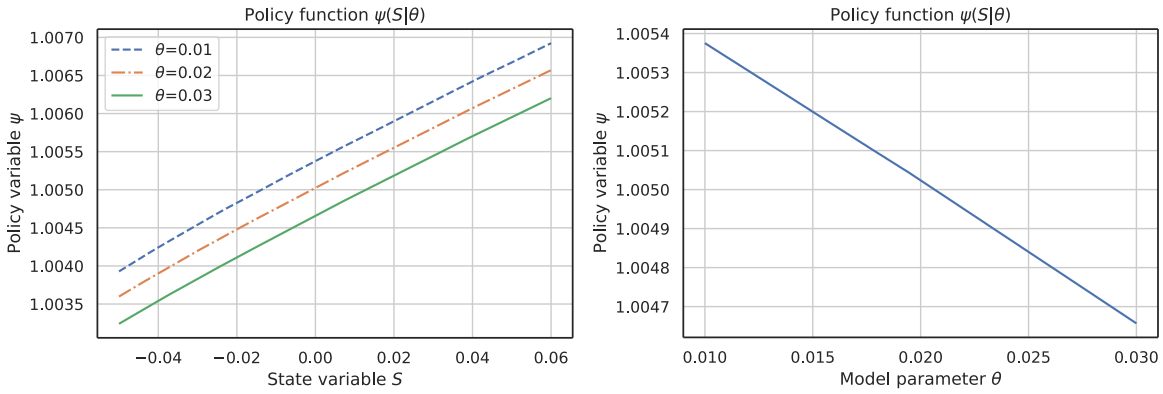


Figure 1.3: Extended neural network representation that captures the mapping from state variables as well as model parameters to the output variables. The left plot varies the state variable and displays the policy functions for selected parameters. The right plot fixed the state variables at $S = 0$ and varies the model parameter.

20. 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.

21. In practice, it was helpful to increase the size of the neural networks to approximate the additional complexity arising through adding parameters as pseudo state variables.

Neural Network-Based Particle Filter. The second important step is to evaluate the fit of the model with the data in an efficient manner. The observation equation that connects the state variables with the observables $\mathbb{Y}_t$ can be written as:

$$\mathbb{Y}_t = g(\mathbb{S}_t|\tilde{\Theta}) + u_t, \tag{1.28}$$

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

We use a particle filter to extract the hidden states and shocks due to the nonlinearity of the solution (Fernández-Villaverde and Rubio-Ramírez 2007; Herbst and Schorfheide 2015)[22]. The particle filter gives us then the likelihood of the model:

$$\mathscr{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right). \tag{1.29}$$

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 thousands of times, which would render estimation infeasible.

To overcome this second 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:

$$\mathscr{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) = \Omega^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right), \tag{1.30}$$

where $\mathscr{L}$ is the likelihood and the function Ω^{PF} is unknown. 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 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^{PF}_{NN} that gives us directly the output of the particle filter:

$$\mathscr{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) = \Omega^{PF}_{NN}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \tag{1.31}$$

where $\mathscr{L}$ is the likelihood of the model and Ω^{PF}_{NN} is the neural network associated with the particle filter.

22. The particle filter has been used to filter highly nonlinear models with for instance occasionally binding constraints (Gust et al. 2017; Atkinson, Richter, and Throckmorton 2020) or (endogenous) multiple equilibria (Aruoba, Cuba-Borda, and Schorfheide 2013; Rottner 2021). However, the particle filter has not been applied to large HANK models so far.

This approach differs on one important margin to the standard approach of using filters. The standard approach evaluates 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 few thousand 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. Conceptually, this can be seen as a nonlinear interpolation of the likelihood values using neural networks.

To train this separate neural network, we create a dataset of parameter values and corresponding likelihoods that we obtained from the particle filter.[23] We split the sample 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 $\Omega_{NN}^{PF}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right)$, the likelihood of the model can be evaluated at a specific draw for negligible costs. Combined with the extended neural network, this allows to generate millions of draws for large and complex methods.[24] An important additional advantage of this method is that the neural network removes some of the noise associated with the particle filter.

A graphical characterization of the neural network based particle filter method can be seen in Figure 1.4. 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. The graph also shows how the neural network removes the noise associated with the particle filter

Bayesian Estimation We are now equipped 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:

$$p\left(\tilde{\Theta}|\mathbb{Y}_{1:T}\right) \propto \mathscr{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \times p(\tilde{\Theta}). \tag{1.32}$$

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.[25]

23. Importantly, the extended neural network of the model is used to calculate likelihood with the particle filter.

24. Importantly, the calculation of the likelihood can be parallelized so that a larger amount of values as input for the neural network can be calculated.

25. 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. Furthermore, such lower and upper bounds are not strictly necessary.

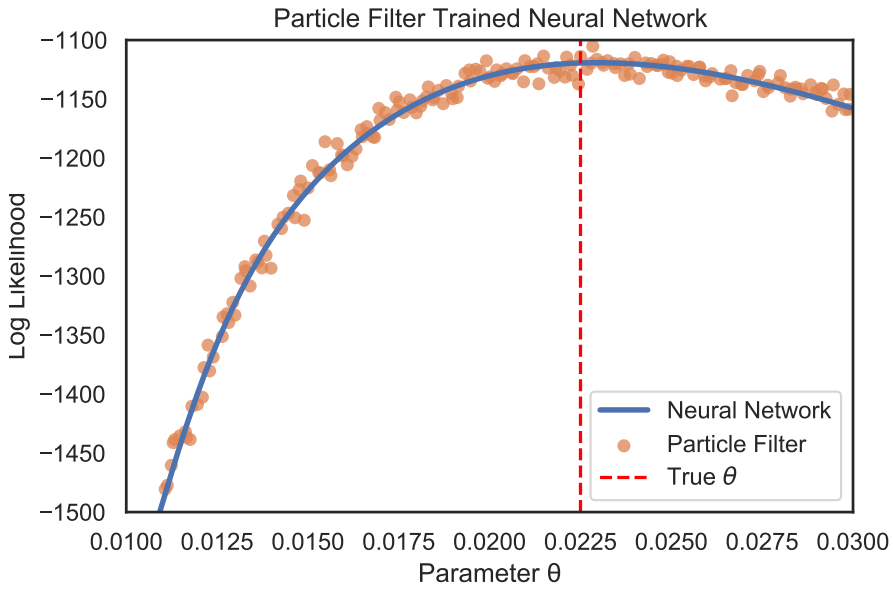


Figure 1.4: Neural network based particle filter method that captures the mapping from the parameter to the log likelihood. The orange dots represent the data sample, where the log likelihood value has been calculated with the particle filter. The blue line is the neural network, which was trained with these data points from the particle filter. The red dashed line indicates the true value.

We can construct the posterior distribution with a Random Walk Metropolis Hastings algorithm to find the posterior of the model. However, our setup requires an important change in the algorithm. We first solve the model using the extended neural network. Then, we train the neural network based particle filter to obtain the likelihoods. After this step, we start to draw from the proposal density. Importantly, it takes us then only milliseconds per a parameter draw since we have precomputed all relevant objects.

1.5 Solving and Estimating a Nonlinear Heterogeneous New Keynesian 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 parameters. We also emphasize the importance of the connection between idiosyncratic and aggregate risk, which underlines the necessity of establishing such a nonlinear approach.

1.5.1 Model

The model is a medium scale HANK model that captures idiosyncratic and aggregate risk. The first key ingredient is heterogeneity with households facing 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.

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:

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

where ζ_t is an aggregate preference shock, which follows an AR(1) process $\zeta_t = \rho_\zeta \zeta_{t-1} + \varepsilon_t^\zeta$. C_t is aggregate consumption.[26] The budget constraint in real terms can be written as:

$$C_t^i + B_t^i = W_t s_t^i H_t^i + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i - T_t^i + Div_t^i, \tag{1.33}$$

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. The agents individual labor productivity s_t^i is stochastic and follows an AR(1) process in logs $s_t^i = \rho_s s_{t-1}^i + \varepsilon_t^{s,i}$. The agents face a borrowing limit $\underline{b}$, which implies:

$$B_t \geq \underline{B}. \tag{1.34}$$

The first order conditions can be written as

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{\zeta_{t+1}}{\zeta_t} \right) \left(\frac{\lambda_t^i}{\lambda_{t+1}^i} \right)^\sigma \frac{1}{\Pi_{t+1}} \right] + \mu_t^i, \tag{1.35}$$

$$\lambda_t = C_t^i - hC_{t-1} \qquad \chi (H_t^i)^\eta = (C_t^i)^{-\sigma} s_t^i W_t \tag{1.36}$$

where $\mu_t^i \geq 0$ is the normalized multiplier on the individual borrowing limit in equation (1.34).

26. Auclert, Rognlie, and Straub (2020) discuss the role of habit formation for the marginal propensity to consume.

Firms

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

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

$$Y_t = \left(\int_0^1 (Y_t^j)^{\frac{\varepsilon-1}{\varepsilon}} df \right)^{\frac{\varepsilon}{\varepsilon-1}}, \tag{1.37}$$

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:

$$P_t = \left(\int_0^1 (P_t^j)^{1-\varepsilon}) \right)^{\frac{1}{1-\varepsilon}}, Y_t^j = \left(\frac{P_t^j}{P_t} \right)^{-\varepsilon} Y_t. \tag{1.38}$$

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:

$$Y_t^j = Z_t N_t^j, \tag{1.39}$$

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

$$Z_t = g_t Z_{t-1} \tag{1.40}$$

where the trend growth rate is subject to idiosyncratic shocks

$$g_t = \bar{g} \exp(\varepsilon_t^g) \tag{1.41}$$

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

$$W_t = Z_t MC_t. \tag{1.42}$$

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

$$\max_{P_t^j} P_t^j \left(\frac{P_t^j}{P_t} \right)^{-\varepsilon} \frac{Y_t}{P_t} - MC_t \left(\frac{P_t^j}{P_t} \right)^{-\varepsilon} - \frac{\varphi}{2} \left(\frac{P_t^j}{\Pi P_{t-1}^j} - 1 \right)^2 Y_t, \tag{1.43}$$

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:

$$\left[\varphi^R\left(\frac{\Pi_t}{\Pi}-1\right)\frac{\Pi_t}{\Pi}\right]=(1-\varepsilon)+\varepsilon MC_t+\varphi^R\mathbb{E}_t\frac{\Pi_{t+1}}{R_t}\left[\left(\frac{\Pi_{t+1}}{\Pi}-1\right)\frac{\Pi_{t+1}}{\Pi}\frac{Y_{t+1}}{Y_t}\right], \quad (1.44)$$

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

$$Div_t = Y_t - W_t Y_t. \quad (1.45)$$

The dividends are distributed equally among the households so that $Div_t = Div_t^i$.[27]

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. In addition to this, there are i.i.d. monetary policy shocks mp_t. The zero lower bound restricts the nominal interest rate. The rule is given as:

$$R_t^N = \left(R_{t-1}^N\right)^{\rho_R}\left(R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{Y_t}{Z_t Y}\right)^{\theta_Y}\right)^{1-\rho_R}\exp(mp_t), \quad (1.46)$$

$$R_t = \max\left[1, R_t^N\right]. \quad (1.47)$$

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

$$D_t = \frac{R_{t-1}}{\Pi_t}D_{t-1} - T_t. \quad (1.48)$$

27. An alternative formulation would be to make the dividend payments depending on the individual productivity of the agent along the lines of Kaplan, Moll, and Violante (2018).

Market Clearing

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

$$N_t = \int N_t^j dj = \int s_t^i H_t^i di \tag{1.49}$$

$$D_t = \int B_t^i di \tag{1.50}$$

$$Y_t = \int C_t^i di. \tag{1.51}$$

1.5.2 Calibration

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, Melosi, and Rottner (2019) as well as the heterogeneous agent frameworks of Kaplan, Moll, and Violante (2018) and Fernández-Villaverde et al. (2021). We also target some key moments of US data using the horizon 2000:Q1 to 2019:Q4. Table 1.1 provides an overview of the calibration.

We set the discount factor β to 0.9975, which implies a 100 basis points contribution to the annualized real interest rate. We set the quarter to quarter average growth rate g to 0.3% in line with US GDP per capita data from 2000:Q1 until 2019:Q4. Combined with the discount factor, this results in real interest rate of 2.2%. The disutility of labor supply is chosen to have a labor supply of 1 in the case of deterministic steady state of the representative agent economy. Consumption habit h is set to a conventional value. The inverse Frisch labor elasticity φ is based on Chetty et al. (2011). The individual borrowing limit is set to -0.29 as in Fernández-Villaverde et al. (2021), which corresponds roughly to a one month labor income. The price elasticity of demand ε targets a market power of 10%. The Rotemberg adjustment costs is set to 1000 in line with the literature. Government debt is set to 0.25 as in Fernández-Villaverde et al. (2021). The inflation target of the central bank Π corresponds to a net inflation rate of 2% per annum. The output target corresponds to the detrended deterministic steady state of the representative agent economy. The monetary policy responds to inflation deviations ϕ_Π and output deviations ϕ_Y, which are set to standard values with 2 and 0.5, respectively. The persistence of the Taylor rule ρ_R is calibrated to 0.25.

Aggregate and Individual Shocks. We set the standard deviations of the aggregate shocks to match PCE core inflation, real per capita GDP growth and the effective federal funds rate.[28] We set the persistence of the preference shock ρ_ζ to 0.85 so that the average inflation

28. Our model predicts a standard deviation for quarterly real GDP per capita growth of 0.6 (data: 0.6), for annualized PCE core inflation of 0.5 (data: 0.5) and the federal funds rate of 2.6 (data: 1.9).

Table 1.1: Calibration for Data-Generating Process

Parameters	Sign	Value	Target/Source
Discount factor	β	0.9975	Real interest rate
Average growth rate	g	1.003	Real GDP growth
Risk aversion	σ	1	Log utility
Disutility of labor	χ	0.86	SS labor supply
Consumption habit	h	0.5	Conventional
Inverse Frisch labor elasticity	φ	0.75^{-1}	Chetty et al. (2011)
Borrowing limit	$\underline{B}$	-0.29	Fernández-Villaverde et al. (2021)
Government debt	D	0.25	Fernández-Villaverde et al. (2021)
Price elasticity of demand	ε	11	Market power
Rotemberg adjustment costs	φ^R	1000	Slope of NKPC
Inflation target	Π	1.005	Inflation target
Output target	Y	1	Deterministic steady state output
Persistence taylor rule	ρ_R	0.25	Conventional
Inflation response	ϕ_π	2.0	Conventional
Output response	ϕ_Y	0.5	Conventional
Persistence labor productivity	ρ_s	0.8	Fernández-Villaverde et al. (2021)
Std. dev. labor productivity	σ_s	2.25%	Share of borrowers
Persistence preference shock	ρ_ζ	0.85	Mean inflation rate
Std. dev. preference shock	σ_ζ	2.5%	Std. dev. inflation
Std. dev. growth rate shock	σ_g	0.6%	Std. dev. GDP growth
Std. dev. MP Shock	σ_{mp}	0.4%	Std. dev. federal funds rate

corresponds to its data equivalent of 1.72. The persistence of the individual labor productivity shock ρ_s follows Fernández-Villaverde et al. (2021). The standard deviation is chosen to match a more usual moment that is related to the wealth distribution. We target that share of borrowers to be around 30% in line with Kaplan, Moll, and Violante (2018).

1.5.3 Neural Networks Based Solution and Estimation

We estimate the nonlinear HANK model with our newly developed neural network based Bayesian methods that are based on the extended neural network and the neural network based particle filter. To provide a controlled environment, the calibrated model is used as the true-data generating process.

We base the analysis on three standard time series that cover a period of 500 periods, which are generated with the calibrated model. Output growth, annualized quarterly inflation rate and the annualized quarterly interest rate are our observables. While we choose for the demonstration a parsimonious setup, the approach could easily extended to contain variables related to the distribution such as selected quantiles of the wealth distribution.

The observation equation is given as

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 100 \ln\left(\frac{\tilde{Y}_t g_t}{\tilde{Y}_{t-1}}\right) \\ 400 \ln\left(\Pi_t\right) \\ 400 \ln\left(R_t\right) \end{bmatrix} + u_t \tag{1.52}$$

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 likelihood of the particle filter. As in Gust et al. (2017), the variance of the measurement error for each time series is a fraction m_E of its variance so that $\Sigma_u = m_E \times \text{diag}\left(V[\mathbb{Y}_{1:T}]\right)$.[29]

The estimation focuses on four key parameters: the standard deviation of the three aggregate shocks as well as the standard deviation of the individual labor productivity shock. The prior distributions are truncated normal densities.[30] The truncation ensures that the drawn parameters lie inside the bounds that have been imposed while solving the extended neural network.[31] The prior mean μ corresponds to the true value, while the standard deviation σ is loose. The bounds a and b are set to allow for a variety of potential values. Table 1.2

29. To be consistent with the formulation of the particle filter, we combine our data series with some measurement error.

30. The probability density function of the truncated normal is $f(x;\mu,\sigma,a,b,) = \frac{1}{\sigma} \frac{\phi((x-\mu)/\sigma)}{\Phi((b-\mu)/\sigma) - \Phi((a-\mu)/\sigma)}$

31. While a truncated prior density is helpful, this is not necessary. The extended neural network can be solved over a distribution without bounds and can also to some extent extrapolate.

Table 1.2: Prior Distributions

Parameter	Distribution	Mean μ	SD σ	Lower Bound a	Upper Bound b
σ_s	Truncated Normal	2.25%	0.01	1%	3%
σ_ζ	Truncated Normal	2.5%	1%	1%	3%
σ_g	Truncated Normal	0.6%	1%	0.1%	1.2%
σ_{mp}	Truncated Normal	0.4%	1%	0.1%	0.9%

summarizes the priors. To evaluate the posterior, a Random Walk Metropolis Hastings algorithm evaluates 1 million draws after a burn-in of 100000 draws.

Neural Networks. We set up two neural networks for the policy functions. One neural network is for the aggregate policy functions, while the other one targets the individual policy functions. The input of both neural networks are the parameters to be estimated and the state variables. Each neural network contains 4 hidden layers with 128 neurons each. The activation function for the hidden layers are parametric Rectified Linear Unit (PReLU). We use 100000 iterations to ensure convergence of the algorithm. After each iteration, the economy is simulated for 20 periods. The batch size is set to 500. The distribution of agents is approximated with 100 agents, that is $L = 100$. We use one deep neural network for the particle filter. This additional neural network consists of 3 layers with 32 neurons each. The activation functions for the hidden layers are Sigmoid Linear Unit (SiLU). We calculate the particle filter at 600 drawn parameter points.[32]

Comparison with Classical Solution Methods. As neural network based solution techniques are very new, we want to compare our results for the calibrated model with classical solution methods. Unfortunately, the nonlinear HANK model is so complex so that we restrict ourselves to the representative version for this exercise. This enables us to compare the results with a classical solution method. We choose a time iteration algorithm that uses piecewise linear policy functions following Richter, Throckmorton, and Walker (2014), which is often applied in the context of New Keynesian models with a zero lower bound.

We compare the impulse response functions of the three aggregate shocks for both solution methods. Figure 1.5 shows the dynamics for the preference shock. The Figure highlights the dynamics are very similar and that there are only small discrepancies.[33] The impulse response

32. We draw from a shuffled Sobol sequence to generate those points. Alternatively, we could have drawn random points. The particle filter itself has 1000 particles.

33. It is not surprisingly to have small deviations as both procedures approximate the solution. The first major difference of the two methods is the usage of a different class of function for the approximation of the policy functions (neural networks vs piecewise linear functions). The second major difference is the approximation of

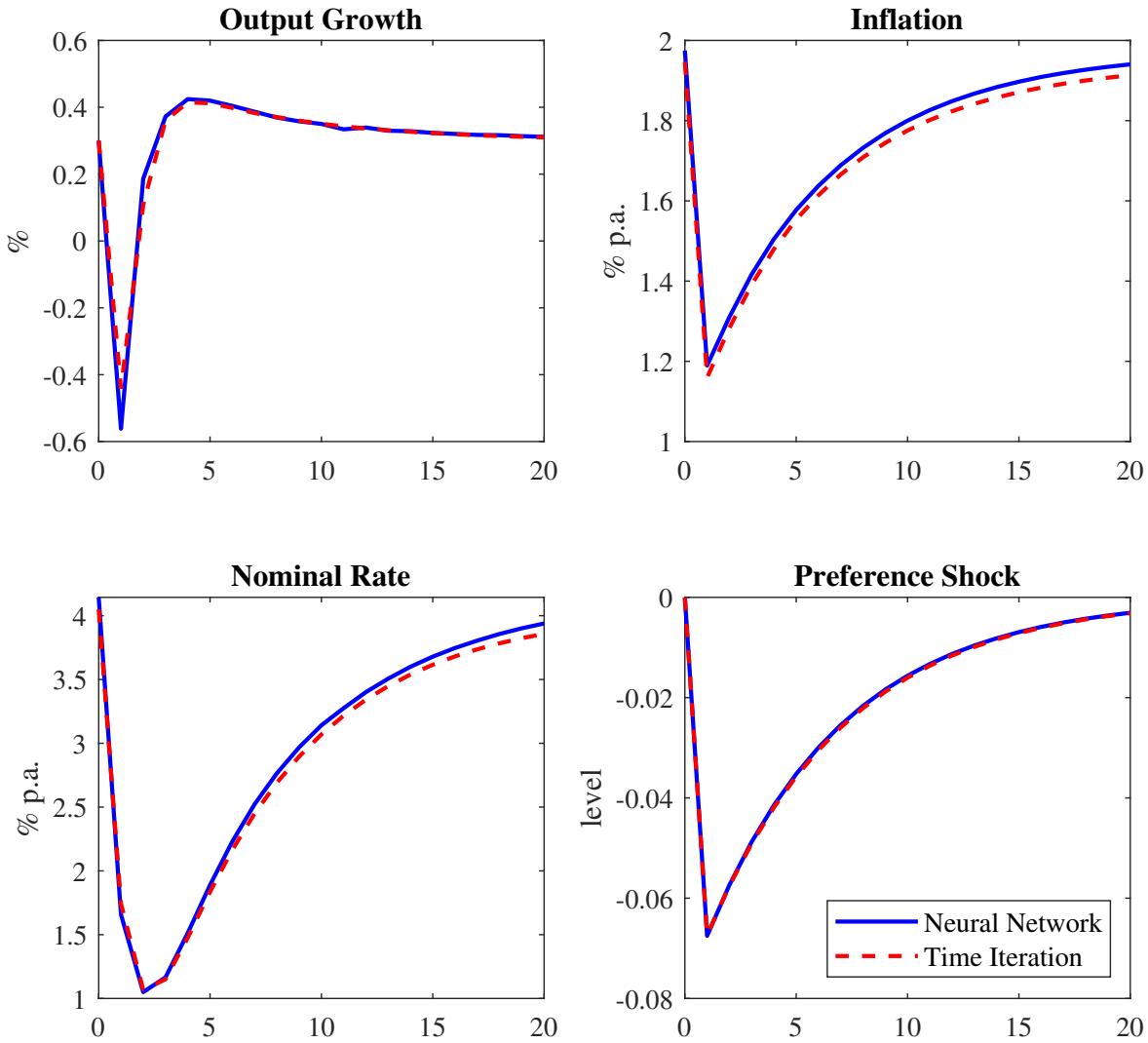


Figure 1.5: Comparison of neural network based solution method with time iteration. Impulse response function for a preference shocks.

functions for the growth rate shock and monetary policy shock are also very similar as can be seen in Appendix 1.D. This comparison verifies that our solution procedure successfully captures the equilibrium conditions of the economy for the representative agent version.

1.5.4 Mapping the model in the General Neural Network Based Estimation Framework

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:

$$\mathbb{S}_t = \left\{ \left\{ \tilde{B}^i_{t-1} \right\}_{i=1}^{L}, \left\{ s^i_t \right\}_{i=1}^{L}, R^N_{t-1}, \tilde{C}_{t-1}, \zeta_t, g_t, mp_t \right\} \tag{1.53}$$

$$\nu_t = \left\{ \left\{ \varepsilon^{s,i}_t \right\}_{i=1}^{L}, \varepsilon^{\zeta}_t, \varepsilon^{g}_t, \varepsilon^{mp}_t \right\} \tag{1.54}$$

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

$$\psi_t = \left\{ \left\{ \tilde{C}^i_t \right\}_{i=1}^{L}, \left\{ N^i_t \right\}_{i=1}^{L}, \left\{ \tilde{B}_t \right\}_{i=1}^{L}, \left\{ \mu_t \right\}_{i=1}^{L}, \tilde{T}_t, \tilde{Y}_t, \tilde{Div}_t, MC_t \right\} \tag{1.55}$$

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

$$\bar{\Theta} = \left\{ \beta, \sigma, \chi, h, \varphi, \underline{B}, D, \varepsilon, \varphi^R, g, \Pi, Y, \rho_r, \kappa_\Pi, \kappa_Y, \rho_s, \rho_\zeta \right\} \tag{1.56}$$

$$\tilde{\Theta} = \left\{ \sigma_s, \sigma_\zeta, \sigma_g, \sigma_{mp} \right\} \tag{1.57}$$

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:

$$\begin{pmatrix} \left\{ N^i_t \right\}_{i=1}^{L} \\ \left\{ \mu^i_t \right\}_{i=1}^{L} \end{pmatrix} = \left\{ \psi^I_{NN} \left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta} | \bar{\Theta} \right) \right\}_{i=1}^{L}, \tag{1.58}$$

expectations. While the neural network based approaches uses the all-in-expectation operator, the time iteration code uses Gauss-Hermite quadrature for expectations.

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:

$$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi_{NN}^A \left(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta}\right) \tag{1.59}$$

1.6 Results

In this section, we present the results of the neural network based Bayesian estimation for the nonlinear HANK model. Afterwards, we show how the zero lower bound affects the results.

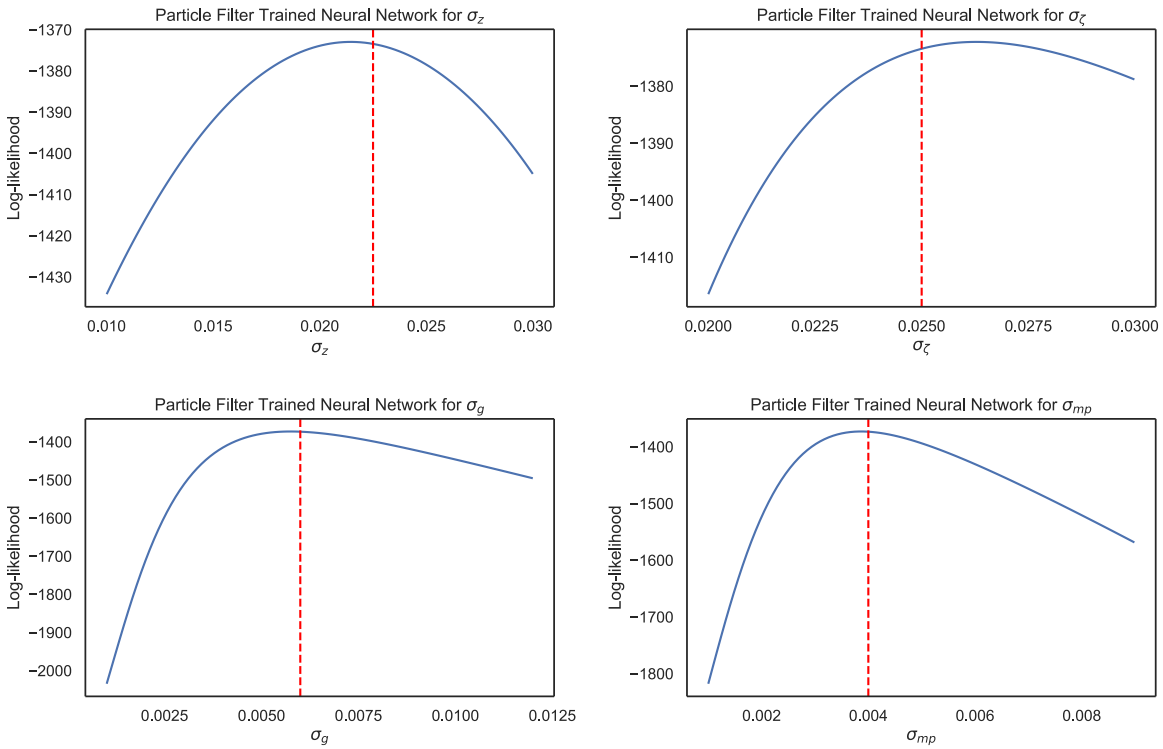


Figure 1.6: Particle filter trained neural network for the nonlinear HANK model. his trained neural networks capture how the estimated parameters affect the likelihood. Each plot varies the parameter of interest, while keeping the other parameters at their true values.

1.6.1 Neural Network Based Bayesian Estimation

We begin with the results for the particle filter trained neural network, which are displayed in Figure 1.6. This trained neural networks capture how the estimated parameters affect the likelihood. Each plot varies the parameter of interest, while keeping the other parameters at their true values. We can see how the likelihood peaks close to its true value. Furthermore,

we can see the difference in the steepness of the likelihood function. This shows that we have a different precision for the different parameters. It is instructive to assess these charts as this function is the basis for the Metropolis Hastings algorithm.

Table 1.3: Posterior Distribution

Parameter	Posterior Median	5%	95%
σ_s	2.14%	1.96%	2.51%
σ_ζ	2.66%	2.50%	2.81%
σ_g	0.58%	0.54%	0.62%
σ_{mp}	0.39%	0.37%	0.41%

To obtain the posterior distribution, we use a Metropolis Hastings algorithm. Table 1.3 summarizes the results of the Bayesian estimation. We can see that the posterior median is close to the true value and it is contained in the 90% confidence interval. Nevertheless, the posterior median for the standard deviation of the idiosyncratic shock is slightly below its true value, while the opposite characteristic can be seen for the demand shock. The reason for this slight deviation is threefold. First, we have added a measurement error to the data in line with the particle filter, which complicates the estimation. Second, we have limited the length of the observed time series for the estimation.[34] Third, the likelihood, especially for the demand shock, is rather flat. As a consequence, adding a measurement error and having only limited amount of data affects the results. A potential solution to this problem could be to add a additional observable variables to increase the precision.

Even though there is a small deviation of the posterior and true value, this shows nevertheless that the method can be used to recover the true data generating process. Importantly, this implies that the neural network based Bayesian estimation can be applied for very complex models such as a nonlinear HANK model.

1.6.2 Aggregate Nonlinearities

While we have shown that our approach can estimate a nonlinear models, we now also want to emphasize the importance of aggregate nonlinearities. For this reason, we compare two versions of our HANK model. Our baseline model features the zero lower bound, while we mute this aggregate nonlinearity for a counterfactual model.

Figure 1.7 compares the ergodic distribution of the models. This highlights that the zero lower bound, which occurs wth a probability slightly over 10%, affects the ergodic

34. We could either extend this timeline or use several draws in a Monte-Carlo study.

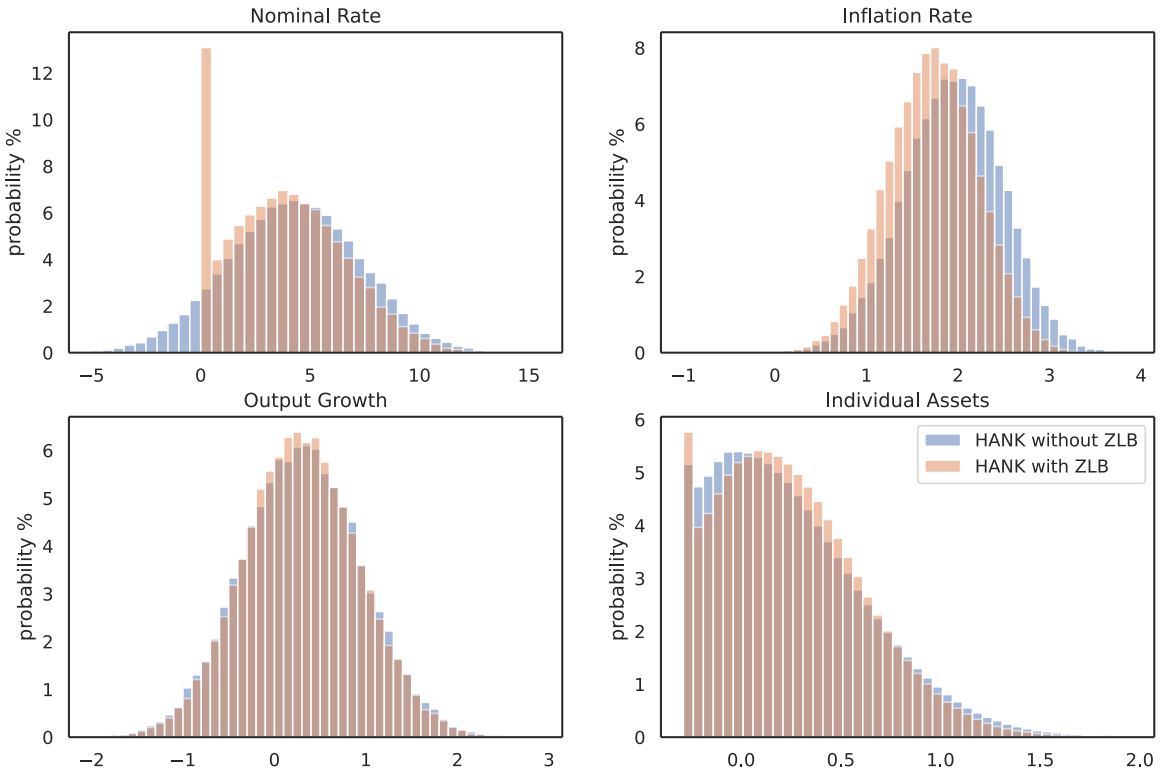


Figure 1.7: Comparison of the ergodic distribution of the nonlinear HANK model with and without the zero lower bound. The models are evaluated at the posterior median. Units: Annualized inflation rate and nominal rate, quarterly rate for output growth.

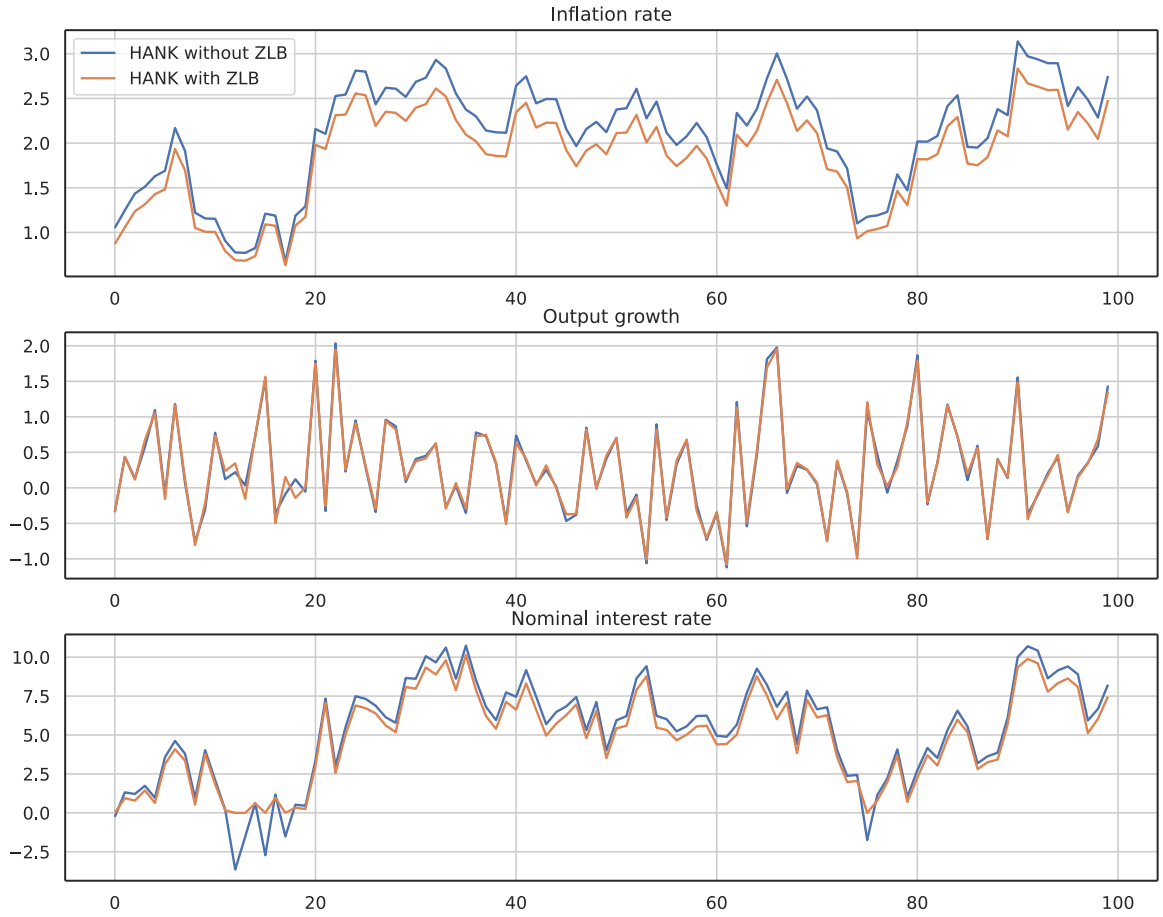


Figure 1.8: Comparison of path with and without the zero lower bound for nonlinear HANK model. . The models are evaluated at the posterior median. Units: Annualized inflation rate and nominal interest rate, quarterly rate for output growth.

distribution. First of all, we can see that the average inflation rate is now sufficiently lower and the distribution is more shifted to the right. The reason is that the zero lower bound creates deflationary pressure. Importantly, this also depends on the amount of heterogeneity as outlined in Fernández-Villaverde et al. (2021). An increased idiosyncratic uncertainty requires a lower market clearing interest rate as the agents want to insure against their labor income risk. Therefore, the idiosyncratic risk pushes the nominal interest rate down and makes a zero lower bound episode more likely. At the same time, the zero lower bound also affects the distribution of assets. In particular, it increases the amount of agents that are at the borrowing limit. The impact on output growth of the zero lower bound is rather muted as this is dominated by the trend component.

Figure 1.8 shows how this affects the evolvement of a time series. Using the same shock sequence, we simulate the model with and without the zero lower bound. This again emphasizes that the zero lower bound lowers the nominal interest rate and creates downward pressure on inflation. As the simulated data series is influenced by the aggregate nonlinearities, it is important to account for this in an estimation.

1.7 Conclusion

In this paper, we outline a novel neural-networks Bayesian estimation procedure, which can be used to estimate models that have been out of reach so far. Our approach uses neural networks in a twofold manner. First, we use an extended neural network that includes state variables and parameters simultaneously. This allows us to solve for different parameter combinations in a single step and enables the estimation of large models. Second, we train an additional neural network to approximate the results of the particle filter.

We apply our techniques to the estimation, in a laboratory setting, of a nonlinear HANK model featuring idiosyncratic and aggregate risk simultaneously. We show that our method can recover the true data-generating process. In the future, we will integrate microdata in the estimation of the HANK model and further increase its complexity.

The proposed neural-networks-based Bayesian 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.

Appendix 1.A Neural Network-Based Solution Algorithm for HANK

The algorithm uses a neural network approach to approximate policy functions based on Maliar, Maliar, and Winant (2021)[35]. 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

   $$\begin{pmatrix} \{N^i_t\}^L_{i=1} \\ \{\lambda^i_t\}^L_{i=1} \end{pmatrix} = \left\{\psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}^L_{i=1} \tag{1.60}$$

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

   $$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{1.61}$$

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

   $$\{N^i_t\}^L_{i=1}, \{\lambda^i_t\}^L_{i=1}, \Pi_t, \tilde{W}_t \tag{1.62}$$

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

   $$\tilde{T}_t = \left(\frac{1}{L}\sum_{i=1}^{L} B^i_{t-1} R_{t-1}\right)\frac{1}{\Pi_t g_t} \tag{1.63}$$

   $$N_t = \left(\frac{1}{L}\sum_{i=1}^{L} N^i_t s^i_t\right) \tag{1.64}$$

   $$\tilde{Y}_t = N_t \tag{1.65}$$

   $$\tilde{\psi}_t = \tilde{W}_t \tag{1.66}$$

   $$\tilde{Div}_t = \tilde{Y}_t - \tilde{W}_t N_t \tag{1.67}$$

35. They use this approach to solve a consumption saving problem and Krusell-Smoth economy.

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

$$R_t^N = \left(R_{t-1}^N\right)^{\rho_R} \left(R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_\Pi} \left(\frac{\tilde{Y}_t}{\tilde{Y}} \right)^{\theta_Y} \right)^{1-\rho_R}, \tag{1.68}$$

$$R_t = \max\left[1, R_t^N\right] \tag{1.69}$$

We pursue with calculating for each household individual variables:

$$\left\{ \tilde{C}_t^i = \left[\frac{s_t^i \tilde{W}_t}{\chi (H_t^i)^\eta} \right]^{\frac{1}{\sigma}} \right\}_{i=1}^{L} \tag{1.70}$$

$$\left\{ \omega_t^i = \tilde{W}_t s_t^i N_t^i + \frac{B_{t-1}^i R_{t-1}}{\Pi_t} - \tilde{T}_t + \tilde{Div}_t \right\}_{i=1}^{L} \tag{1.71}$$

$$\left\{ B_t^i = \omega_t^i - C_t^i \right\}_{i=1}^{L} \tag{1.72}$$

Aggregate consumption is given as

$$C_t = \frac{1}{L} \sum_{i=1}^{L} C_t^i \tag{1.73}$$

We use the all-in-one expectation method of Maliar, Maliar, and Winant (2021), which uses two randomly drawn shocks for each AR(1) process to evaluate the expectations:

$$\text{Random Draws 1:} \left\{ \varepsilon_t^{\zeta,1} \right\}_{i=1}^{L}, \varepsilon_t^{\zeta,1} \tag{1.74}$$

$$\text{Random Draws 2:} \left\{ \varepsilon_t^{\zeta,2} \right\}_{i=1}^{L}, \varepsilon_t^{\zeta,2} \tag{1.75}$$

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

$$\ln(\zeta_{t+1}^1) = \rho_\zeta \ln(\zeta_t) + \varepsilon_{t+1}^{\zeta,1} \tag{1.76}$$

$$\ln(s_{t+1}^1) = \rho_s \ln(s_t) + \varepsilon_{t+1}^{s,1} \tag{1.77}$$

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:

$$\begin{pmatrix} \{N_{t+1}^{i,1}\}_{i=1}^{L} \\ \{\lambda_t^{i,1}\}_{i=1}^{L} \end{pmatrix} = \psi_1 \left(\mathbb{S}_{t+1}^1; \Theta \right), \tag{1.78}$$

and similarly for the aggregate control variables:

$$\begin{pmatrix} \Pi_t^1 \\ \tilde{W}_t^1 \end{pmatrix} = \psi_2 \left(\mathbb{S}_{t+1}^1; \Theta \right) \tag{1.79}$$

We can now calculate the aggregate variables again

$$T_{t+1}^1 = \left(\frac{1}{L} \sum_{i=1}^{L} B_t^i R_t \right) \frac{1}{\Pi_{t+1}^1} \tag{1.80}$$

$$N_{t+1}^1 = \left(\frac{1}{L} \sum_{i=1}^{L} N_{t+1}^{i,1} s_{t+1}^{i,1} \right) \tag{1.81}$$

$$Y_{t+1}^1 = A N_{t+1}^1 \tag{1.82}$$

$$Div_{t+1}^1 = Y_{t+1}^1 - W_{t+1}^1 N_{t+1}^1 \tag{1.83}$$

We pursue with calculating for each household individual variables:

$$\left\{ C_t^i = \left[\frac{s_{t+1}^{i,1} W_{t+1}^1}{\chi (H_{t+1}^{i,1})^{\eta}} \right]^{\frac{1}{\sigma}} \right\}_{i=1}^{L} \tag{1.84}$$

$$\left\{ \omega_{t+1}^{i,1} = W_{t+1}^1 s_{t+1}^{i,1} N_{t+1}^{i,1} + \frac{B_t^i R_t}{\Pi_{t+1^1}} - T_{t+1}^1 + Div_{t+1}^1 \right\}_{i=1}^{L} \tag{1.85}$$

$$\left\{ B_{t+1}^{i,1} = \omega_{t+1}^{i,1} - C_{t+1}^{i,1} \right\}_{i=1}^{L} \tag{1.86}$$

Aggregate consumption is given as

$$C_{t+1}^1 = \frac{1}{L} \sum_{i=1}^{L} C_{t+1}^{i,1}. \tag{1.87}$$

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 $t+1$.

$$\left\{R_1^{i,1} = \beta R_t \left[\left(\frac{\zeta_{t+1}^1}{\zeta_t}\right)\left(\frac{C_t^i}{C_{t+1}^{i,1}}\right)^{\sigma}\frac{1}{\Pi_{t+1}^1}\right] - \lambda_t^i\right\}_{i=1}^{L} \tag{1.88}$$

$$\left\{R_2^{i,1} = \Psi^{FB}\left(B_t^i + \underline{B}, 1-\lambda_t^i\right)\right\}_{i=1}^{L} \tag{1.89}$$

$$R_N^1 = \left[\varphi\left(\frac{\Pi_t}{\Pi}-1\right)\frac{\Pi_t}{\Pi}\right] - (1-\varepsilon) - \varepsilon MC_t - \varphi\frac{\Pi_{t+1}^1}{R_t}\left[\left(\frac{\Pi_{t+1}^1}{\Pi}-1\right)\frac{\Pi_{t+1}^1}{\Pi}\frac{Y_{t+1}^1}{Y_t}\right] \tag{1.90}$$

$$R_M^1 = D - \frac{1}{L}\sum_{i=1}^{L} B_t^i \tag{1.91}$$

$$R_{MN}^1 = D - \frac{1}{L}\sum_{i=1}^{L} B_{t+1}^{i,1} \tag{1.92}$$

$$R_R^1 = \tilde{Y}_t - C_t \tag{1.93}$$

$$R_{RN}^1 = Y_{t+1}^1 - C_{t+1} \tag{1.94}$$

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

$$\left\{R_1^{i,2}\right\}_{i=1}^{L}, \left\{R_2^{i,2}\right\}_{i=1}^{L}, R_N^2, R_M^2, R_{MN}^2, R_R^2, R_{RN}^2 \tag{1.95}$$

3. Define the loss function:

$$R^2 = \sum_{i=1}^{L}\alpha_1^i R_1^{i,1} R_1^{i,2} + \sum_{i=1}^{L}\alpha_2^i R_2^{i,1} R_2^{i,2} + \alpha_N R_N^1 R_N^2 + \alpha_M R_M^1 R_M^2 + \tag{1.96}$$

$$\alpha_{MN} R_{MN}^1 R_{MN}^2 + \alpha_R R_R^1 R_R^2 + \alpha_{RN} R_{RN}^1 R_{RN}^2 \tag{1.97}$$

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^{sim} 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:

$$e \geq 0, \;\; f \geq 0, \;\; e \times f = 0 \tag{1.98}$$

The Fischer-Burmeister function is defined as

$$\Psi^{FB}(e,f) = e + f - \sqrt{e^2 + f^2} \tag{1.99}$$

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 (1.34)). In the algorithm, we used λ_t^i, which is defined as follows:

$$\lambda_t^i = 1 - \mu_t^i \tag{1.100}$$

The complementary slackness conditions can be written as

$$1 - \lambda_t^i \geq 0 \tag{1.101}$$

$$\left(B_t^i - \underline{B}\right) \geq 0 \tag{1.102}$$

$$\left(1 - \lambda_t^i\right) \times \left(B_t^i - \underline{B}\right) > 0 \tag{1.103}$$

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

$$\Psi^{FB}\left(1 - \lambda_t^i, B_t^i - \underline{B}\right) \tag{1.104}$$

Appendix 1.B 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)

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:

$$L = \Omega_{PF}\left(\tilde{\Theta}|Data\right) \tag{1.105}$$

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

$$R^2 = (L - L^v)^2 \tag{1.106}$$

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

Appendix 1.C Detrended Equilibrium Conditions

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

$$\tilde{C}_t^i + \tilde{B}_t^i = W_t s_t^i \tilde{H}_t^i + \frac{R_{t-1}}{\Pi_t g_t} \tilde{B}_{t-1}^i - \tilde{T}_t^i + \tilde{Div}_t^i, \tag{1.107}$$

$$\tilde{\lambda}_t = \tilde{C}_t - h\frac{\tilde{C}_{t-1}}{g_t} \tag{1.108}$$

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{\zeta_{t+1}}{\zeta_t} \right) \left(\frac{\tilde{\lambda}_t^i}{\tilde{\lambda}_{t+1}^i} \right)^{\sigma} \frac{1}{\Pi_{t+1} g_{t+1}^{\sigma}} \right] + \mu_t^i, \tag{1.109}$$

$$\chi (H_t^i)^{\eta} = (\tilde{\lambda}_t^i)^{-\sigma} \left(s_t^i \tilde{W}_t \left(1 - \tau_t (1-\gamma) \left(s_t^i \tilde{W}_t H_t^i \right)^{-\gamma} \right) \right) \tag{1.110}$$

$$\tilde{Y}_t^j = N_t^j, \tag{1.111}$$

$$\tilde{W}_t = MC_t \tag{1.112}$$

$$\tilde{Div}_t = \tilde{Y}_t - W_t \tilde{Y}_t \tag{1.113}$$

$$\left[\varphi^R \left(\frac{\Pi_t}{\Pi} - 1 \right) \frac{\Pi_t}{\Pi} \right] = (1-\varepsilon) + \varepsilon MC_t + \varphi^R \mathbb{E}_t \frac{\Pi_{t+1}}{R_t} \left[\left(\frac{\Pi_{t+1}}{\Pi} - 1 \right) \frac{\Pi_{t+1} g_t^{\sigma-1}}{\Pi} \frac{\tilde{Y}_{t+1}}{\tilde{Y}_t} \right], \tag{1.114}$$

$$R_t^n = (R_{t-1}^N)^{\rho_R} \left(R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_{\Pi}} \left(\frac{\tilde{Y}_t}{\tilde{Y}} \right)^{\theta_Y} \right)^{1-\rho_R}, \tag{1.115}$$

$$R_t = \left[1, R_t^N \right] \tag{1.116}$$

$$\tilde{D}_t = \frac{R_{t-1}}{\Pi_t g_t} \tilde{D}_{t-1} - \tilde{T}_t \tag{1.117}$$

Appendix 1.D Comparison of Neural Networks Solution Method to Time Iteration

We compare the impulse response functions of the neural network solution method to a classical solution method (time iteration). Figure 1.9 reports the dynamics for growth rate shock and Figure 1.10 shows the case for the monetary policy shock.

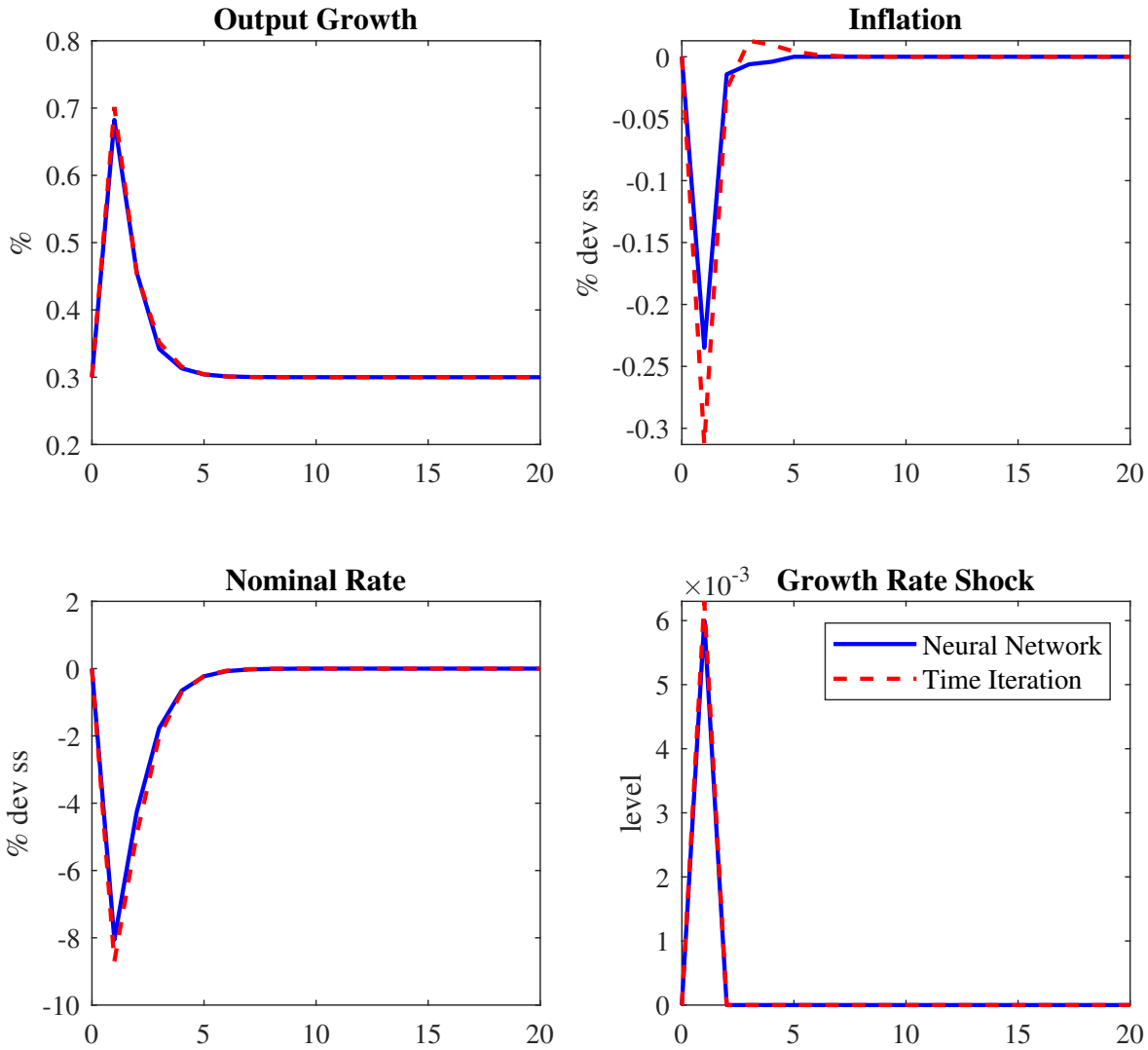


Figure 1.9: Comparison of neural network based solution method with time iteration. Impulse response function for a growth rate shock.

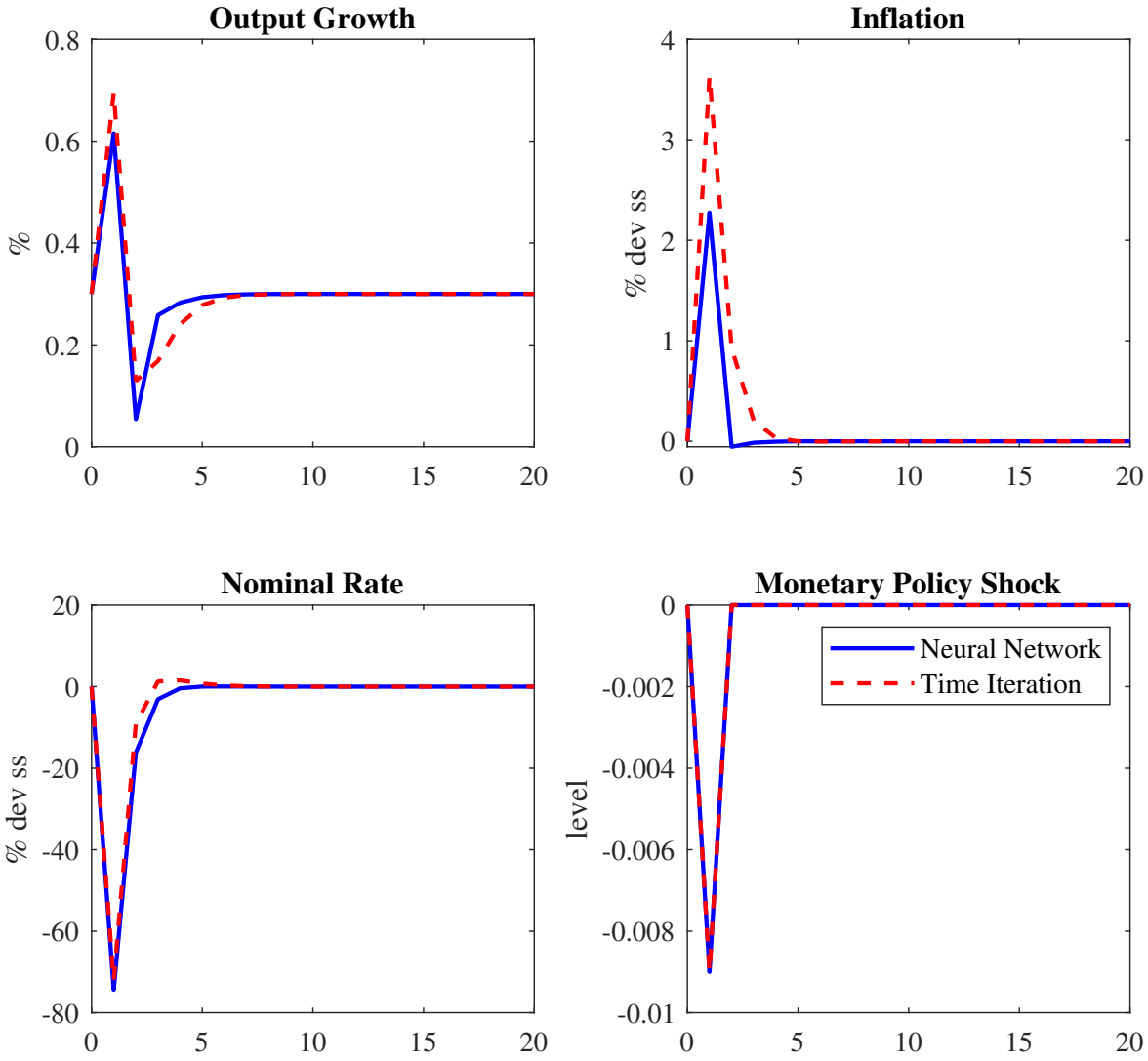


Figure 1.10: Comparison of neural network based solution method with time iteration. Impulse response function for a monetary policy shock.

References

Aruoba, S., P. Cuba-Borda, and F. Schorfheide. 2013. *Macroeconomic Dynamics Near the Zero Lower Bound: A Tale of Two Countries.* NBER working paper 19248.

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

Auclert, A., B. Bardóczy, M. Rognlie, and L. Straub. 2019. *Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models.* Technical report. National Bureau of Economic Research.

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

Azinovic, M., L. Gaegauf, and S. Scheidegger. 2019. "Deep equilibrium nets." *Available at SSRN 3393482.*

Bach, F. 2017. "Breaking the curse of dimensionality with convex neural networks." *The Journal of Machine Learning Research* 18 (1): 629–681.

Barron, A. R. 1993. "Universal approximation bounds for superpositions of a sigmoidal function." *IEEE Transactions on Information theory* 39 (3): 930–945.

Bayer, C., B. Born, and R. Luetticke. 2020. "Shocks, frictions, and inequality in US business cycles."

Bianchi, F., L. Melosi, and M. Rottner. 2019. *Hitting the elusive inflation target.* Technical report. National Bureau of Economic Research.

Chen, H., A. Didisheim, and S. Scheidegger. 2021. "Deep Structural Estimation: With an Application to Option Pricing." *Available at SSRN.*

Chetty, R., A. Guren, D. Manoli, and A. Weber. 2011. "Are micro and macro labor supply elasticities consistent? A review of evidence on the intensive and extensive margins." *American Economic Review* 101 (3): 471–75.

Cybenko, G. 1989. "Approximation by superpositions of a sigmoidal function." *Mathematics of control, signals and systems* 2 (4): 303–314.

Duarte, V. 2018a. *Gradient-Based Structural Estimation.* Technical report. Working Paper.

———. 2018b. *Machine learning for continuous-time finance.* Technical report. Working Paper.

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

Fernández-Villaverde, J., G. Nuño, G. Sorg-Langhans, and M. Vogler. 2020. "Solving High-Dimensional Dynamic ProgrammingProblems using Deep Learning."

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

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

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

Guerrieri, L., and M. Iacoviello. 2015. "OccBin: A toolkit for solving dynamic models with occasionally binding constraints easily." *Journal of Monetary Economics* 70:22–38.

Gust, C., E. Herbst, D. López-Salido, and M. E. Smith. 2017. "The Empirical Implications of the Interest-Rate Lower Bound." *American Economic Review* 107 (7): 1971–2006.

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

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

Kaplan, G., B. Moll, and G. L. Violante. 2018. "Monetary policy according to HANK." *American Economic Review* 108 (3): 697–743.

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

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

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

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

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

Maliar, L., S. Maliar, and P. Winant. 2021. "Deep learning for solving dynamic economic models." *Journal of Monetary Economics*.

Norets, A. 2012. "Estimation of dynamic discrete choice models using artificial neural network approximations." *Econometric Reviews* 31 (1): 84–106.

Richter, A. W., N. A. Throckmorton, and T. B. Walker. 2014. "Accuracy, speed and robustness of policy function iteration." *Computational Economics* 44 (4): 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."

Villa, A. T., and V. Valaitis. 2019. "Machine learning projection methods for macro-finance models." *Economic Research Initiatives at Duke (ERID) Working Paper*.

Chapter 2

Limits on Mortgage Lending

2.1 Introduction

After the financial crisis, that was accompanied by a collapse in house prices and rise in mortgage defaults, many countries in Europe have introduced macroprudential regulation that target mortgage lending in order to improve financial stability. On the other hand, the opponents are concerned about the effect those policies may have on homeownership rate, especially among younger households. This is particularly salient in countries, such as Germany, with one of the lowest homeownership rates in Europe. Nevertheless, in 2017, the German Federal Financial Supervisory Authority (BaFin) was granted power to introduce loan-to-value (LTV) limit and minimum amortization requirements for residential real estate loans (BaFin 2017).

In this paper I study the potential impact of the commonly used borrower-based macroprudential limits over the life cycle. Namely, I consider limits on loan-to-value (LTV), debt-to-income (DTI), payment-to-income (PTI), and minimum amortization requirement. For this purpose, I build a life-cycle model with incomplete markets, housing, and long-term mortgage[1]. The households make decisions about renting or owning a house, consumption, savings, mortgage borrowing, and default. The house prices, rents, and mortgage premiums are determined endogenously. One of the non-standard features of the model is that the banks can screen lenders[2], but are only able to condition the interest rate on LTV ratio at origination. The model is calibrated to match key moments from Household Finance and Consumption Survey (HFCS) data on Germany. In the baseline solution the limits are not binding for borrowing decisions of households. For the policy experiments, I consider tightening each of

1. I build most on the work of Kaplan, Mitman, and Violante (2020), Karlman, Kinnerud, and Kragh-Sørensen (2020), and Hu (2021).

2. The screening technology is modelled similarly to Hu (2021).

the limits one at a time and study their effects on homeownership rates over the life-cycle and the long-run welfare.

I find that even very strict limits on mortgage lending have modest impact on homeownership rates. Only tightening of the LTV limit leads to a significant decrease in homeownership rate among younger households. However, the welfare impact of the more stringent LTV limit is positive. In an economy without LTV limit, the households tend to overborrow, leading to higher house prices. Therefore, stricter LTV limits help to correct this externality. Same cannot be said about income-based measures, DTI and PTI limits. Due to the limited pricing ability of the banks, tighter DTI and PTI limits can lead to lower interest rates, more mortgage borrowing, higher house prices, and overall a negative welfare effect. However, the impact of these limits is small in magnitude.

This paper is related to a large body of literature that studies the impact of borrower-based macroprudential instruments in heterogeneous-agent incomplete-markets models. However, most of the literature does not explicitly focus on homeownership rate over the life cycle. Also, very few papers consider as many borrower-based macroprudential instruments. There are handful of papers using life-cycle models to study German housing market. Kaas et al. (2021) studies determinants of low homeownership rate in Germany. Using similar model, Grevenbrock (2019) studies the effect of coresidency decision, comparing homeownership rates in Germany and Italy. Neither of them focuses on borrower-based macroprudential limits.

The rest of the paper is organized as follows. Section 2.2 introduces the model, followed by the discussion of the calibration to the German economy in Section 2.3. Then, I examine the results of the policy experiments in Section 2.4 and conclude in Section 2.5.

2.2 Model

The model is based on a standard incomplete markets life-cycle model with housing and long-term mortgage. I deviate from the standard model in three aspects: lenders can screen borrowers, borrowers are pooled according to LTV ratio at origination, and mortgage payments follow the annuity payment schedule.

Even in countries where there are no legally binding macroprudential limits on mortgage lending, there are relatively few loans with high LTV, DTI, or PTI ratios. To match this observation, the banks in this model are equipped with a loan screening technology as in Hu (2021). Only loans to borrowers with probability of default under certain threshold are approved.

In the standard framework, the banks are assumed to set the interest rate to make zero profits on a loan-by-loan basis. I assume that lenders can condition the mortgage interest rate only on the LTV ratio at origination[3]. For each pool of borrowers, that have the same LTV at origination, the banks are assumed to set the mortgage interest rate so that they make zero profits in equilibrium. The pooling of borrowers leads to additional mechanism through which macroprudential limits can affect households. Changes in regulatory limits can affect the composition of borrowers in the LTV pools, which in turn will have an effect on the equilibrium mortgage interest rates.

I explicitly keep track of the number of mortgage payments made to make sure the model reflects accurately the annuity repayment schedule used for most mortgage loans. In the literature, mortgage loans are usually assumed to have fixed maturity, and they are amortized over the remaining life or until certain age T. This saves a state variable, as the maturity of a mortgage is the difference between current age j and T. However, it leads to counterfactually long maturities for younger households or mortgage payment schedules that are more spread out compared to the standard annuity payment schedule. This difference could make PTI limits for younger households less binding.

In this model households make several discrete choices for tenure, house size, mortgage size, and default. For all of discrete choices I have added taste shocks for two reasons. First, to generate default. In this type of models, forcing households to default would require unreasonably high maintenance costs or a depreciation shock. Instead, I include a large taste shock for the discrete decisions on whether to continue paying the mortgage, sell the house, or default. The second reason is that the taste shocks make the model smoother, allowing for more efficient solution. In the description of the model, I omit the taste shocks for simplicity of notation.

2.2.1 Households

Households enter the economy at age J_{start}. Over the lifetime they face idiosyncratic income shocks until they retire at age J_{retire}. There is no mortality risk during life and death arrives with certainty at the age J_{end}. Households derive utility from consumption good c and housing services h according to a CRRA utility function in Equation (2.1).

$$u_j(c,h_j) = e_j \frac{\left(c^{\alpha} h^{1-\alpha}\right)^{1-\sigma}}{1-\sigma} \tag{2.1}$$

3. This is computationally very convenient, as I am already keeping track of the LTV at origination to calculate the mortgage payments and amount owed.

where σ is the coefficient of relative risk aversion and α governs the weight of consumption in the utility. Changes in the household size over the life-cycle are captured by exogenous equivalence scale e_j.

In the last period of life, households get additional utility from leaving bequest W' according to the warm-glow bequest function. This gives households an incentive to die with positive net wealth, as in the data.

$$v(W') = B\frac{(W')^{1-\sigma}}{1-\sigma} \tag{2.2}$$

Each period, household i receives idiosyncratic endowment $y_{i,j}$.

$$y_{i,j} = \chi_j \cdot \varepsilon_i \tag{2.3}$$

where χ_j is a deterministic age profile of income that is common for all households and ε_i is a persistent idiosyncratic component of income that follows a finite-state first-order Markov process with realizations and transition matrix in Equation (2.4).

$$E = (\varepsilon_1, \ldots, \varepsilon_N), \quad \Pi = \begin{bmatrix} \pi(\varepsilon_1, \varepsilon_1) & \ldots & \pi(\varepsilon_1, \varepsilon_N) \\ \vdots & \ddots & \vdots \\ \pi(\varepsilon_N, \varepsilon_1) & \ldots & \pi(\varepsilon_N, \varepsilon_N) \end{bmatrix} \tag{2.4}$$

Households have access to two assets: one period risk-free bond and a house. The only form of debt available to households is a long-term mortgage collateralized by a house.

Households can buy one period risk-free bonds b for a price $p_b = \frac{1}{1+r_b}$. The interest rate r_b is given exogenously.

In order to consume housing services, households can choose to rent or buy a house. In both cases the size of the house is chosen form a set $\mathscr{H} = \{h_1, \ldots, h_N\}$. Houses provides housing services equal to their size.

Depreciation of a house is assumed to be completely offset by maintenance payments made by the owner of the house. For an owner-occupied house the maintenance cost is a fraction δ_h of the size of the house. The maintenance cost of a rented house is paid by the rental company and is a fraction $\delta_r > \delta_h$ of the size of the rented house.

Moving out of an owned house is costly: fraction θ of the sold house is lost as various transaction fees. However, changing the size of a rented housing is costless for the renter.

The supply of housing in the economy is fixed; house and rental prices are determined endogenously[4].

Purchase of a house can be financed with a long-term mortgage, using the house as a collateral. Instead of choosing mortgage amount directly, households pick a loan-to-value ratio from a grid $\Lambda = \{\lambda_1 = 0.0, \lambda_2 = 0.5, \ldots, \lambda_N = 1.0\}$. The initial amount borrowed is $A = \lambda p_h h$.

The mortgage is repaid following an annuity schedule. The maturity of the mortgage cannot be longer than $\bar{M}$ years and the loan has to be fully repaid before the end of life. Therefore the maximum maturity of the mortgage for a household aged j is $\bar{M}_j = min(\bar{M}, J_{end} - j)$.

In addition to h and λ, I keep track of the number of mortgage payments already made, denoted by n. Therefore, the maturity of the mortgage at origination for a household aged j who has made n payments is $N_j(n) = min(\bar{M}, n + J_{live} - j)$. A single mortgage payment, including both interest and principal, is given by Equation (2.5).

$$P_j(h, \lambda, n) = \frac{r_m(\lambda)(1 + r_m(\lambda))^{N_j(n)}}{(1 + r_m(\lambda))^{N_j(n)} - 1} A \tag{2.5}$$

where $r_m(\lambda)$ is the mortgage interest rate, that depends on the chosen loan-to-value ratio. The mortgage interest rate stays the same throughout the duration of the mortgage[5].

Remaining mortgage principal can be calculate according to Equation (2.6).

$$m_j(h, \lambda, n) = P_j(h, \lambda, n) \left(\frac{1}{r_m(\lambda)} - \frac{(1 + r_m(\lambda))^{n - N_j(n)}}{r_m(\lambda)} \right) \tag{2.6}$$

At the origination, the loan has to comply with LTV, DTI, and PTI limits.

$$\lambda \leq LTV_{limit}, \quad \frac{\lambda p_h h}{y_{i,j}} \leq DTI_{limit}, \quad \frac{P_j(h, \lambda, n)}{y_{i,j}} \leq PTI_{limit} \tag{2.7}$$

2.2.2 Rental firm

Rental price p_r is determined on a competitive rental market. The representative rental firm solves the following recursive optimization problem in Equation (2.8) as in Karlman,

4. Initial amount of housing in the economy is determined by setting $p_h = 1.0$ and allowing the housing supply to be perfectly flexible as in Karlman, Kinnerud, and Kragh-Sørensen (2020). After which the supply of housing is fixed and the price will adjust endogenously.

5. I have omitted the dependence of the loan amount and interest rate on the time of taking the mortgage, because in the steady-state both mortgage interest rate and the house price are constant across time. However, it matters during transitions where the mortgage interest rate and the house price at the origination of the mortgage can be recovered based on the number of payments made n.

Kinnerud, and Kragh-Sørensen (2020).

$$J(\tilde{H}) = \max_{\tilde{H}'} \left(p_r \tilde{H}' - p_h(\tilde{H}' - \tilde{H}) - \delta_r \tilde{H} + \left(\frac{1}{1+r_b} \right) J(\tilde{H}') \right) \tag{2.8}$$

where $\tilde{H}$ is the amount of housing the rental firm owns, p_r is the rental price, δ_r is the maintenance cost and the rental firm discounts the future at rate r_b that is same as the risk-free interest rate on one period bonds.

The solution to the rental firms optimization problem leads to closed form solution for the rental price given by Equation (2.9).

$$p_r = \left(\frac{1}{1+r_b} \right) \left(r_b p_h + p_h - p_h' + \delta_r \right) \tag{2.9}$$

2.2.3 Banking sector

Mortgages are provided by competitive risk neutral lenders. The banks have a simple loan screening technology as in Hu (2021). The loans are only granted to borrowers with a default probability under a certain exogenous threshold Ψ.

I assume that the lenders are only able to condition the interest rate on borrowers' LTV ratio at origination. They price the mortgage with an interest rate $r_m(\lambda)$ such that they break even in every LTV category. Hence, there is a cross-subsidization in mortgage pricing.

This restriction can arise, for example, from internal procedures of the bank; it is simpler to communicate to the client interest rate that only depends on few easy-to-understand indicators. Moreover, Best et al. (2020), using the UK data, provides evidence that mortgage interest rate is close to a step function of LTV ratio at origination.

2.2.4 Households' optimization problems

Household enters a period as either (a) a renter, or (b) a homeowner.

(a) A renter can choose whether to continue renting or buy a house. His value function is given by:

$$V_j^R(y,b) = \max\{V_{j,Rent}^R(\cdot), V_{j,Buy}^R(\cdot)\} \tag{2.10}$$

(1) When he decides to continue renting he chooses consumption c, next period bond holdings b', and house to rent $h_r \in H$ to solve the following maximization

problem:

$$\begin{aligned} V^R_{j,Rent}(y,b) = &\max_{c,b',h_r} \left\{ u_j(c,h_r) + \beta \mathbb{E}_{y'|y} V^R_{j+1}(y',b') \right\} \\ &\text{s.t} \quad c + \frac{1}{1+r_b} b' + \underbrace{p_r h_r}_{\text{rent}} \leq y_j + b \\ &\quad b' \geq 0,\ c > 0 \end{aligned} \tag{2.11}$$

(2) When he decides to buy a house he chooses consumption c, next period bond holdings b', a house to buy h', and a loan-to-value ratio for the mortgage λ'. He will start next period as a homeowner.

$$\begin{aligned} V^R_{j,Buy}(y,b) = &\max_{c,b',h',\lambda'} \left\{ u_j(c,h') + \beta \mathbb{E}_{y'|y} V^O_{j+1}(y',b',h',\lambda',n') \right\} \\ &\text{s.t} \quad c + \frac{1}{1+r_b} b' + \underbrace{p_h h'}_{\text{house}} + \underbrace{\delta_h h'}_{\text{maint.}} \leq y_j + b + \underbrace{(1-\eta)\lambda' h'}_{\text{mortgage - fee}} \\ &\quad \lambda' \leq \text{LTV}_{limit} \\ &\quad P_j(h',\lambda',n') \leq \text{PTI}_{limit} y_j \\ &\quad \lambda' p_h h' \leq DTI_{limit} y_j \\ &\quad b' \geq 0,\ c > 0,\ n' = 1 \end{aligned} \tag{2.12}$$

(b) A homeowner has following choices: continue paying the mortgage; sell the house, and repay the mortgage; default, and be forced to rent for a period. The value function of a homeowner is given by:

$$V^O_j(y,b,h,\lambda,n) = \max\{V^O_{j,C}(\cdot), V^O_{j,Sell}(\cdot), V^O_{j,Default}(\cdot)\} \tag{2.13}$$

(1) Homeowner who continues to pay the mortgage payments chooses consumption c and next period bond holdings b' to solve the following maximization problem:

$$\begin{aligned} V^O_{j,C}(y,b,h,\lambda,n) = &\max_{c,b'} \left\{ u_j(c,h) + \beta \mathbb{E}_{y'|y} V^O_{j+1}(y',b',h',\lambda',n') \right\} \\ &\text{s.t} \quad c + \frac{1}{1+r^S} b' + \underbrace{\delta_h h}_{\text{maint.}} + \underbrace{P_j(h',\lambda',n')}_{\text{mortgage pmt.}} \leq y_j + b \\ &\quad h' = h,\ \lambda' = \lambda,\ n' = n+1 \\ &\quad b' \geq 0,\ c > 0 \end{aligned} \tag{2.14}$$

Once the mortgage is fully repaid the loan-to-value is set to $\lambda' = 0.0$, hence there is no further mortgage payments.

(2) Homeowner who decides to sell the old house liquidates all his assets and solves the same problem as a renter, but starting from bond holdings equal to the value of his net assets. Hence he can become a renter or a homeowner[6].

$$V^O_{j,Sell}(y,b,h,\lambda,n) = V^R_j(y,b^h)$$
$$\text{Where} \quad b^h = b + \underbrace{(1-\theta)p_h h}_{\text{sell house}} - \underbrace{P_j(h,\lambda,n) - m_j(h,\lambda,n)}_{\text{repay mortgage}} \tag{2.15}$$

(3) Homeowner can also decide to default, in which case he loses his house and bond holding. Additionally, he spends the period as a renter in the smallest house and faces additional utility penalty Υ that captures the stigma related to the default.

$$V^O_{j,Default}(y,b,h,\lambda,n) = V^R_{j,Rent}(y,b) - \Upsilon \tag{2.16}$$

2.2.5 Equilibrium definition

In this environment, given distribution of age J_{start} households, a stationary competitive equilibrium is a collection of household value functions: $\{V^R_j(y,b), V^R_{j,Rent}(y,b), V^R_{j,buy}(y,b), V^O_j(y,b,h,\lambda,n), V^O_{j,C}(y,b,h,\lambda,n), V^O_{j,Sell}(y,b,h,\lambda,n), V^O_{j,Default}(y,b,h,\lambda,n)\}$, with associated policy functions; prices $\{p_h, p_r, r_b, r_m(\lambda)\}$; quantity of housing H; such that:

1. Given prices $\{p_h, p_r, r_b, r_m(\lambda)\}$, households optimize by solving problems (2.10) to (2.16).
2. The rental price p_r is given by (2.9).
3. The mortgage market clears in each loan-to-value category, with mortgage interest rate given by $r_m(\lambda)$.
4. The total stock of housing H is given by the total demand for housing at price $p_h = 1.0$.
5. Let $s \in \{Rent, Buy, Continue, Sell, Default\}$ be the status of the household and $\omega = \{s,y,b,h,\lambda,n\}$ the state. The distribution of households $\Phi_j(\omega)$ is given by the following

6. This simplification, similar to the one used by Kaplan, Mitman, and Violante (2020), makes the sellers problem computationally much easier.

law of motion for $J_{start} < j < J_{live}$:

$$\Phi_{j+1}(\omega) = \int Q_j(\omega'; \omega) d\Phi_j(\omega)$$

Where $Q(\omega'; \omega)$ is the transition function that defines the probability that a household in a state $\{s, y, b, h, \lambda, n\}$ transitions into a state $\{\omega'\}$.

The model is solved numerically using backwards induction. I have used DC-EGM algorithm by Iskhakov et al. (2017), implemented in PyTorch machine learning framework (Paszke et al. 2017) to benefit from automatic differentiation and GPU computing. For a more detailed discussion on using automatic differentiation in economic models, I refer the reader to Chapter 3.

2.3 Calibration

The model is calibrated to match Household Finance and Consumption Survey (HFCS) data for Germany, with a focus on matching the homeownership rate, wealth, income and mortgage borrowing related characteristics of different age groups. The HFCS dataset contains information about finances of households in 18 euro area countries. The survey was organised in three waves that took place in 2010, 2014 and 2017.

I choose one model period to be equal to five years for two reasons. First, it ensures that when comparing model to the data, there are enough observations in each age category. Second, it reduces dramatically the state space. Since I keep track of both the age and the number of mortgage payments made, choosing five year model period over one year reduces the size of the state space almost by a factor of 25.

To abstract from the educational choices in the early life, households enter the model at age 25, they retire at age 65 and live up until age 80. In the data, I observe that the mortgage borrowing drops significantly in the retirement. Therefore, in the model I allow households to take mortgages until the age of 65. After the retirement, they can still continue repayment of the old mortgages, but cannot get a new one.

The initial distribution of households at age 25 in the model is chosen to match the share of renters and owners in the 25-29 age group in the data. Additionally, I match the bond holdings to the mean liquid wealth.

To estimate the income-age profiles of households I use all three waves available. I follow Lagakos et al. (2018) implementation of Heckman, Lochner, and Taber (1998) approach to separate the age effect from time and cohort effects. This approach exploits the common

Note: The experience profile shows the wages associated with each experience bin relative to the average wage of workers with less than 5 years of experience, which is normalized to 1.

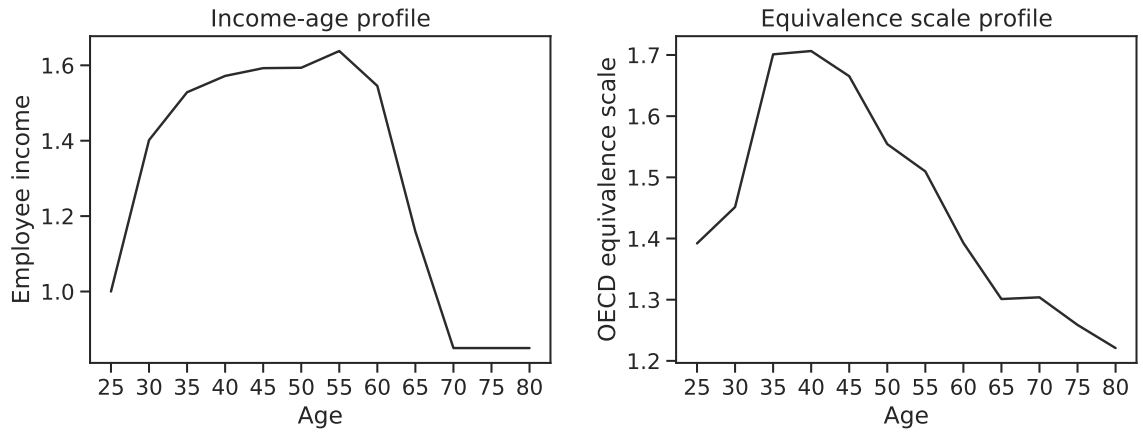
Figure 2.1: Income-experience profile

prediction of many theories on life-cycle wage growth, namely that towards the final years of working life there should be very little wage growth due to age or experience. I use five-year experience bins and assume there is no wage growth due to experience going form 35-39 years of experience to 40+ years. The resulting income-experience profile for Germany is shown in Figure 2.1.

From the estimated income-experience profile I construct the income-age profile as follows. For ages 25 to 55 the income-age profile corresponds to the income-experience profile from 0-4 to 30-34 years of experience. This procedure implicitly assumes that all individuals work around 40 hours a week between ages 25 and 59. In order to construct income at age 60, I multiply the income-experience profile at age 60 by 0.9 proportional to the reduction in hours worked of individuals in this age category. Age 65 marks the retirement age. In order to account for some households delaying their retirement, I use intermediate replacement rate of 75% of age 60 income. For the remaining years until the end of life the replacement rate is 55% of age 60 income. The resulting income-age profiles are shown in Figure 2.2a.

The equivalence scale, which captures changes in household size over the life-cycle, is constructed using the data from the second wage of the HFCS (2014). In particular, I use the variable for consumption units in each household, which is calculated using the OECD scale[7]. I then categorize households into five-year age bins based on age of the head of household.

7. According to the modified OECD scale, the first member of the household is counted as one. Each additional member adds 0.5 when age is over 14 and 0.3 if below 14 (ECB 2020).



(a) **(b)**

Figure 2.2: Life-cycle profiles

Finally, I compute the mean of the consumption units in the households in each age bin. Figure 2.2b shows the resulting equivalence scale profile.

I choose the set of house sizes $\mathscr{H}$ to match the distribution of the value of household's main residence. In particular, the lowest house size corresponds to 20th percentile (approx. €75,000) and the highest - to 80th percentile (approx. €375,000). To transform these values into model units, I divide them by five times the mean annual employee income of households in the 25-29 age bracket (approx. €30,000). I also restrict the house size grid to have five grid points. Thus, $\mathscr{H} \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$.

Calibrated parameter values are presented in Table 2.1 together with their sources or targets.

2.3.1 Model fit

The model fits the overall life-cycle trends in the data reasonably well. Figure 2.3a shows how the model can accurately match the hump shaped homeownership rate profile. This is the main variable of interest for the policy experiments.

The model also matches well the amount of bond holdings of households by age bins, shown in Figure 2.3b. I use the variable on deposits as the data counterpart. I also compare the model fit of bond holdings against alternative variables: financial and net liquid assets. Qualitatively, the shapes of these curves are similar to that of deposits shown in Figure 2.3b. Matching the amount of liquid assets is important, as it determines how easily the households can increase their downpayments in the face of stricter limits on mortgage lending.

Table 2.1: Calibrated parameters

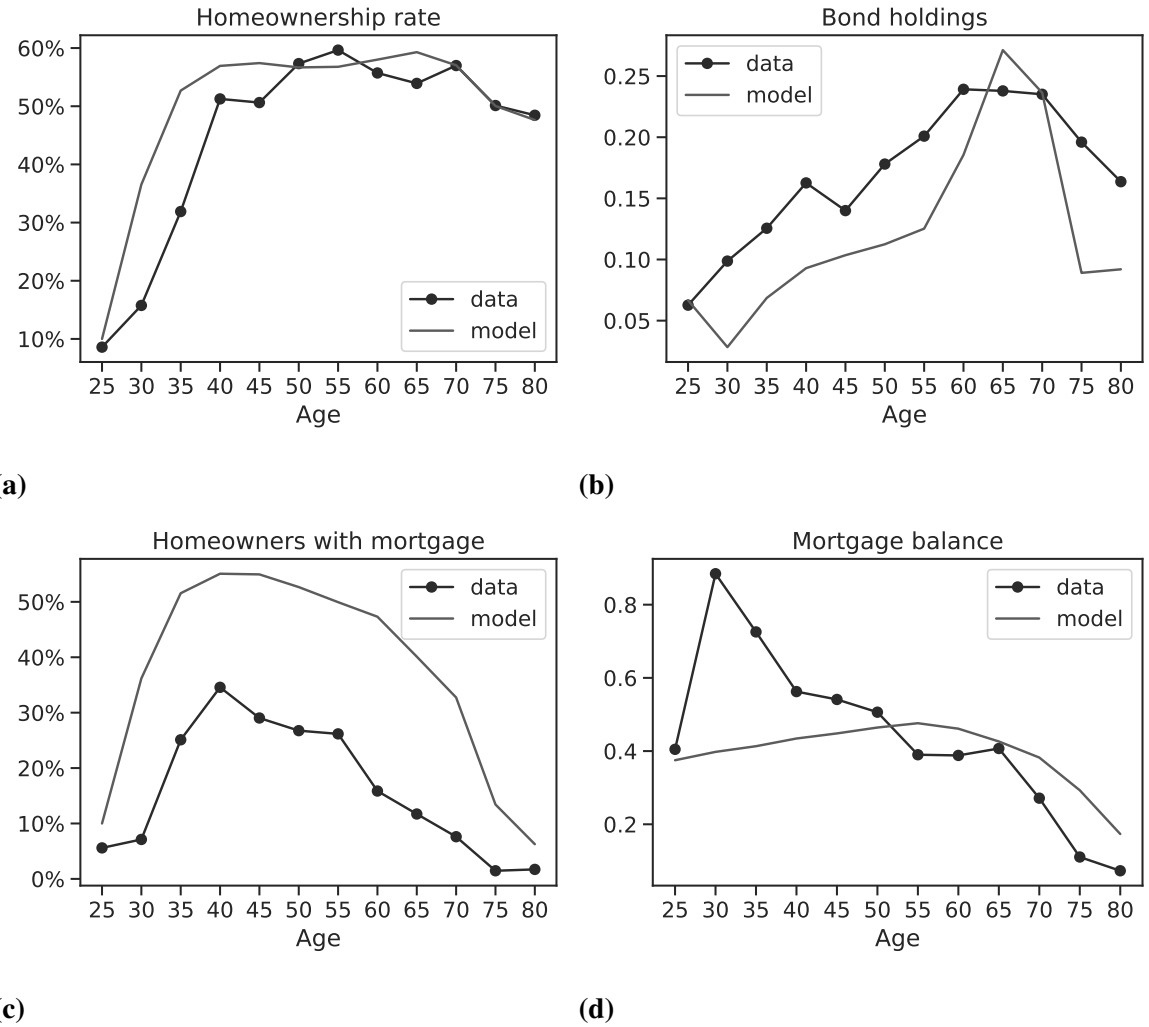
	Description	Value	Source or target
χ_j	Income profile	Figure 2.2a	HFCS (wave 2)
e_j	Equivalence scale	Figure 2.2b	HFCS (wave 2)
$\Phi_{J_{start}}$	Initial dist.		HFCS (wave 2)
$\mathscr{H}$	Set of house sizes	$\{0.5, 1.0, \ldots, 2.5\}$	House value €75k–€375k
σ	Risk aversion	2.0	Standard in literature
α	Expenditure share	0.717	Kaas et al. (2021)
β	Discount factor	0.90	Net worth
B	Bequest weight	0.5	Net worth in retirement
δ_r	Dep. rate, rented	0.05	Homeownership rate
δ_h	Dep. rate, owned	0.02	50-year house lifespan
r_b	Risk-free rate	0.02	Long-term average
$\bar{b}$	Borrowing limit	0.0	
θ	Moving cost	0.137	Kaas et al. (2021)
η	Mortgage fee	0.01	1% of mortgage amount
Υ	Default penalty	1.0	
ρ_ε	Persistence of ε	0.90	
σ_ε	Std. dev. of ε	0.30	
$\mathbb{R}$	Recovery rate	0.70	
Ψ	Screening threshold	0.02	
τ_h	Scale, house	0.1	
τ_d	Scale, decision	0.3	
τ_{LTV}	Scale, LTV	0.1	

Note: Where relevant the model parameter values have been converted to match one year model period. The first block of parameters are either estimated or directly derived from HFCS data. Second block of parameters are set to values found in literature or hand-calibrated to broadly match the data. Third block of parameters control the scale of extreme value type I distributed taste shocks associated with house size, decision (rent, buy or continue, sell, default), and LTV choice.

The model overstates the share of homeowners with mortgage compared to the data. Nevertheless, the shapes of the curves across agges are remarkably similar (see Figure 2.3c). The difference between the curves can be partially explained by the modelling of bequests. Currently, the bequests of the dying households are not redistributed among younger households and are destroyed. Without such redistribution, a larger share of young homeowners have to finance the purchase using mortgage loans.

Figure 2.3d compares the average mortgage loan size by age bins. Overall, the model matches the decreasing pattern of the average mortgage sizes as households get older. Interestingly, the model understates the average mortgage size of the younger households. Which could again be related to the modeling of bequests. Since younger households do not inherit

from older generations, they start from lower house sizes and can only finance payments for smaller mortgages.



Note: The bond holdings from the model are compared with deposits from the data. Both deposits and mortgage balance are normalized by dividing with five times mean annual income at age 25-29, which corresponds to one unit in the model.

Figure 2.3: Model fit

2.4 Policy experiments

I solve the baseline model without imposing restrictions on the mortgage lending. That is, the baseline solution is obtained with LTV limit set at 100%, DTI limit set at 100, and PTI limit set at 100%. The maximum loan maturity in the baseline scenario is 25 years.

For the counterfactual policy experiments, I consider four scenarios focusing on one limit at a time: (1) LTV limit at 70%, (2) DTI limit at 3, (3) PTI limit at 30%, and (4) lowering

maximum maturity to 20 years. This covers the most commonly used borrower-based macroprudential instruments. First three have direct counterparts from the real world, although the exact definition can vary. The last experiment corresponds to tighter amortization requirements. As the loans annuity repayment schedules, shorter maturity will necessarily result in faster amortization of the mortgage principal.

Two of the measures, LTV limit and amortization requirements, are policy instruments currently available to the German financial supervisory authority, BaFin (BaFin 2017). The two other measures, DTI and PTI limits, are not. However, they were considered when the set of instruments were decided upon. Moreover, DTI and PTI limits are effective in many other European countries. Therefore, I have included them in my policy experiments.

I compare the resulting steady states under different policies against the baseline of the model with no borrower-based macroprudential regulation in place. I first consider the effect on the homeownership at different age groups. Then, I compare the welfare effects on a household just entering the economy.

2.4.1 Effect on homeownership rate

The policy experiments I consider are on the stringent side of what is typically implemented in European countries. Nevertheless, their impact on the homeownership rate is not dramatic, as shown in Figure 2.4.

Lowering the LTV limit to 70% reduces the homeownership rate of younger households as their access to mortgage credit is more restricted. Older households, with more accumulated wealth, are less bound by the LTV limit and are able to benefit from lower house prices (see Figure 2.5a).

The effect of income-based measures, DTI and PTI, on homeownership rate is tiny. Surprisingly, in this environment, the tighter DTI and PTI limits lead to slightly higher house prices. Tighter limits on DTI and PTI exclude some borrowers from the mortgage market, who would have otherwise passed the screening by the banks[8]. The remaining borrowers are less likely to default. Since banks are assumed to make zero profits, they will reduce the mortgage interest rate for those LTV categories (see Figure 2.5b). Which in turn leads to more mortgage borrowing by households that pass the DTI and PTI limits, driving up the house prices.

8. Some of the borrowers may opt for lower LTV mortgages. Due to larger downpayment the mortgage amount and subsequent mortgage payments will be smaller, allowing them to comply with the DTI or PTI limits.

Shortening the mortgage maturity from 25 years to 20 years also reduces the homeownership rate among younger households. As the annuity payments for shorter mortgages are higher, the risk that a bad shock will make the household default rises. Therefore, some households no longer pass the screening threshold. In addition, shorter maturity also means faster amortization. Both of these factors reduce the default risk, leading to lower mortgage rates at almost all loan sizes.

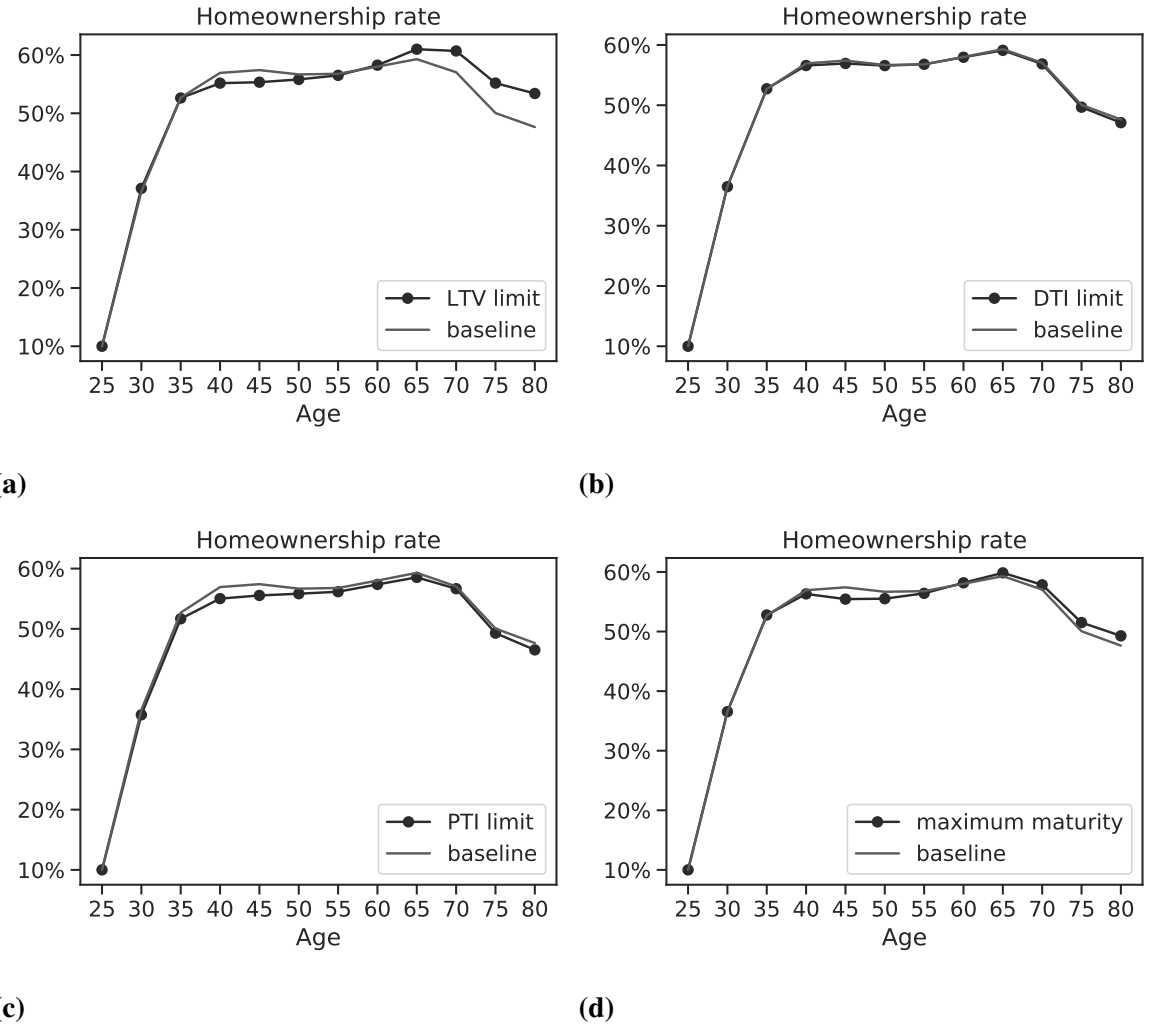
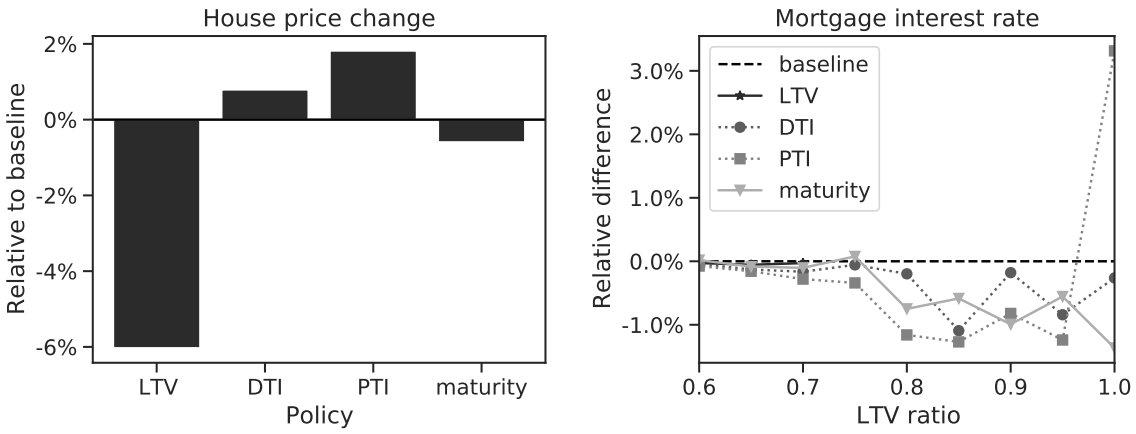


Figure 2.4: Effect on homeownership

2.4.2 Effect on welfare

I measure the effect of different policies on welfare for a household that is entering the economy in consumption equivalent variation. It measures the percentage change in consumption that would make the household indifferent between two stationary economies: baseline economy and one of the four counterfactual economies with limits on mortgage lending.



(a) **(b)**

Figure 2.5: Effect on house price and mortgage interest rates

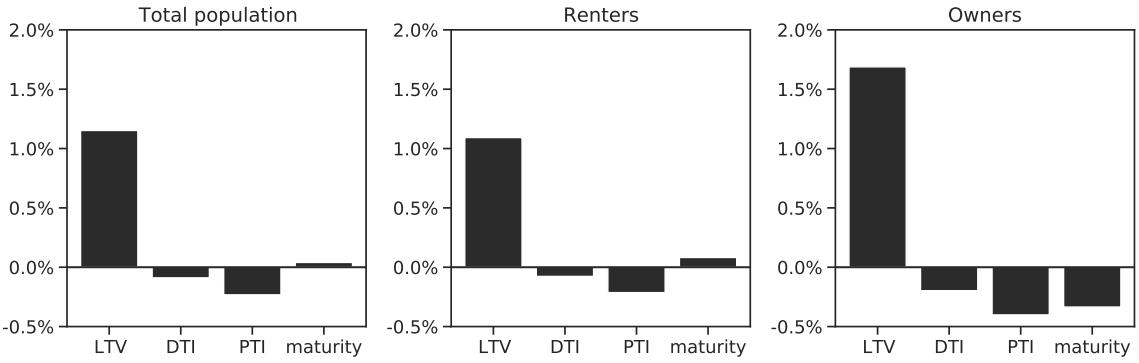


Figure 2.6: Effect on welfare

Out of the four policy experiments considered, LTV limit is the only one that leads to substantial positive welfare effects. Households do not internalize the effect of mortgage borrowing on the house prices, leading to overborrowing and higher house prices. The LTV limit helps to correct this externality. Both renters and homeowners benefit from this policy through lower house and rent prices.

Limits on DTI and PTI ratio have a small negative impact on welfare. As noted earlier, these policies lead to more mortgage borrowing and higher house prices. Thus, exacerbating the overborrowing externality.

Limiting maximum mortgage maturity to 20 years has negligible positive effect on overall welfare. Renters would benefit from this policy through lower rents, implied by lower house prices. On the other hand, the effect on homeowners welfare is slightly negative. This is surprising given that the policy reduces the mortgage rates and house prices. It could stem from the accelerated amortization, which means larger mortgage payments. Therefore some households might find it optimal to buy smaller houses.

2.5 Conclusion

After the financial crisis, that was accompanied by a collapse in house prices and rise in mortgage defaults, many countries in Europe have introduced macroprudential regulation that target mortgage lending in order to improve financial stability. On the other hand, the opponents are concerned about the effect those policies may have on homeownership rate, especially among younger households.

In this paper I study the potential impact of the commonly used borrower-based macroprudential limits over the life cycle. I consider limits on loan-to-value (LTV), debt-to-income (DTI), payment-to-income (PTI), and minimum amortization requirement. To study this question, I build a life-cycle model with incomplete markets, housing, and long-term mortgage. The model is calibrated to match key moments from Household Finance and Consumption Survey (HFCS) data on Germany. In the baseline solution the limits are not binding for borrowing decisions of households. For the policy experiments, I consider tightening each of the limits one at a time and study their effects on homeownership rates over the life-cycle and the long-run welfare.

I find that even very strict limits on mortgage lending have modest impact on homeownership rates. Only tightening of the LTV limit leads to a significant decrease in homeownership rate among younger households. However, the welfare impact of the more stringent LTV limit is positive. In an economy without LTV limit, the households tend to overborrow, leading

to higher house prices. Therefore, stricter LTV limits help to correct this externality. Same cannot be said about income-based measures, DTI and PTI limits. Due to the limited pricing ability of the banks, tighter DTI and PTI limits can lead to lower interest rates, more mortgage borrowing, higher house prices, and overall a negative welfare effect. However, the impact of these limits is small in magnitude.

References

BaFin. 2017. "Residential Mortgage Loans: BaFin Gains New Macro-Prudential Powers." BaFin, 2017. Accessed August 30, 2021. https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Fachartikel/2017/fa_bj_1706_wohnimmobilienkredite_en.html.

Best, M. C., J. S. Cloyne, E. Ilzetzki, and H. J. Kleven. 2020. "Estimating the Elasticity of Intertemporal Substitution Using Mortgage Notches." *The Review of Economic Studies* 87, no. 2 (2020): 656–690.

ECB. 2020. "HFCS User Database Documentation," 190.

Grevenbrock, N. 2019. "Homeownership Rates, Housing Policies and Co-Residence Decisions." *Google Docs.*

Heckman, J. J., L. Lochner, and C. Taber. 1998. *Explaining Rising Wage Inequality: Explorations with a Dynamic General Equilibrium Model of Labor Earnings with Heterogeneous Agents.* Working Paper, Working Paper Series 6384. National Bureau of Economic Research.

Hu, M. 2021. *Default Risk Heterogeneity and Borrower Selection in the Mortgage Market.* SSRN Scholarly Paper ID 3455611. Rochester, NY: Social Science Research Network, 2021.

Iskhakov, F., T. H. Jørgensen, J. Rust, and B. Schjerning. 2017. "The Endogenous Grid Method for Discrete-Continuous Dynamic Choice Models with (or without) Taste Shocks." *Quantitative Economics* 8 (2): 317–365.

Kaas, L., G. Kocharkov, E. Preugschat, and N. Siassi. 2021. "Low Homeownership in Germany—a Quantitative Exploration." *Journal of the European Economic Association* 19, no. 1 (2021): 128–164.

Kaplan, G., K. Mitman, and G. L. Violante. 2020. "The Housing Boom and Bust: Model Meets Evidence." *Journal of Political Economy* 128, no. 9 (2020): 3285–3345.

Karlman, M., K. Kinnerud, and K. Kragh-Sørensen. 2020. "Costly Reversals of Bad Policies: The Case of the Mortgage Interest Deduction." *Review of Economic Dynamics* (2020).

Lagakos, D., B. Moll, T. Porzio, N. Qian, and T. Schoellman. 2018. "Life Cycle Wage Growth across Countries." *Journal of Political Economy* 126, no. 2 (2018): 797–849.

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. 2017. "Automatic Differentiation in Pytorch."

Chapter 3

Backpropagating Through Heterogeneous Agent Models

3.1 Introduction

Heterogeneous agent models are often computationally difficult to solve, and even more difficult to estimate. Yet, this type of models are incredibly useful for addressing questions about inequality and distributional consequences of policies. Coupled with better availability of microdata, it has created a demand for better computational methods.

Over the recent years, computational economics literature has seen an influx of papers introducing methods that build on recent advances in machine learning. At the core of many of those methods are deep neural networks, which are used to parametrize high dimensional policy or value functions as in Maliar, Maliar, and Winant (2021), Azinovic, Gaegauf, and Scheidegger (2019), Duarte (2018), and Fernández-Villaverde et al. (2020) or expectations as in Villa and Valaitis (2019). Deep neural networks allow economists to alleviate the curse of dimensionality and solve increasingly high-dimensional dynamic programming problems.

Deep neural networks can have tens of thousands or even millions of parameters as inputs. The most common way to train a large number of parameters is to use some stochastic gradient descent based optimizer. These optimizers have proven very effective in combination with the backpropagation algorithm (Rumelhart, Hinton, and Williams 1986) that allows calculating the gradient of the objective function very efficiently - approximately with the same computational cost as evaluating the function itself.

This paper explores the application of the backpropagation algorithm in the context of heterogeneous agent models. Unlike the existing literature, I will not introduce a deep neural network to approximate model objects and alleviate the curse of dimensionality. Instead, I will

focus on leveraging the strengths of machine learning software as a general-purpose scientific computing framework that also offers an easy and fast way to use backpropagation. The ability to quickly calculate gradients can aid in clearing markets, calculating transition dynamics, and calibrating or estimating the model. I demonstrate the use of the backpropagation algorithm in each of these applications using a simple model.

In addition, I contribute to the recent literature that highlights the similarities (Igami 2020) and differences (Iskhakov, Rust, and Schjerning 2020) of economic models and structural estimation to the algorithms used in machine learning. Although I don't explicitly introduce a deep neural network, a standard value function iteration algorithm on the grid can be viewed as a recurrent convolutional neural network. This connection is not new for the machine learning literature (Tamar et al. 2017). However, I will make the similarity very clear in the context of the household optimization problem in Section 3.4. The tight connection between the standard value function iteration algorithm and the recurrent convolution neural network is useful because computational economics can benefit from advances, both in hardware and algorithms, in this rapidly progressing field.

The rest of the paper is structured as follows. Section 3.2 explains the backpropagation algorithm. Section 3.3 introduces a simple heterogeneous household model. Section 3.4 illustrates the connection between value function iteration and convolutional neural networks. In Section 3.6 I demonstrate possible applications of the backpropagation algorithm using the simple heterogeneous household model. Finally, Section 3.7 concludes the paper.

3.2 Backpropagation and automatic differentiation

The backpropagation algorithm is a special case of reverse mode automatic differentiation, where the output of a function we are interested in differentiating is a scalar error. This is a typical problem in deep learning as the networks are trained to minimize the distance between predictions and observations. Given the partial derivatives of the error with respect to the parameters of the neural network, an optimizer would take steps adjusting the parameters in a direction that minimizes that distance. In this section, I will give a non-technical explanation of automatic differentiation. I will start from the forward mode automatic differentiation, a conceptually simpler algorithm, and move on to reverse mode automatic differentiation.

Automatic differentiation is a family of techniques to accurately evaluate derivatives of differentiable functions that can be represented by computer programs (Tamar et al. 2017). The main idea is to observe that computer programs are made up of elementary functions. If the elementary function is differentiable, we usually know how to differentiate it. So to find

the derivatives of the main program, representing the function of interest, we can combine the derivatives from the elementary functions using the chain rule.

There are two main modes for automatic differentiation: forward mode and reverse mode. The difference stems from the order of applying the chain rule. The following simple example, adapted from Tamar et al. (2017), will make the distinction clear and helps to build intuition on the strengths and weaknesses of different modes. Consider the following simple function:

$$y = f(x_1, x_2) = x_1 x_2 + ln(x_1)$$

Table 3.1 illustrates how a computer would evaluate the function and how the forward mode automatic differentiation would calculate the derivatives. As the computer evaluates the elementary functions that make up the bigger function, it stores the values in v_i, evaluates the corresponding elementary derivative functions and stores the derivatives in $\dot{v}_i$ as it goes forward. To calculate both partial derivatives, it takes two passes.

Table 3.1: Example of forward mode automatic differentiation

↓	Value	↓	$\frac{\partial y}{\partial x_1}$	$\frac{\partial y}{\partial x_2}$
1	$v_1 = x_1 = 2$	1	$\dot{v}_1 = \frac{\partial v_1}{\partial x_1} = 1$	$\dot{v}_1 = \frac{\partial v_1}{\partial x_2} = 0$
2	$v_2 = x_2 = 5$	2	$\dot{v}_2 = \frac{\partial v_2}{\partial x_1} = 0$	$\dot{v}_2 = \frac{\partial v_2}{\partial x_2} = 1$
3	$v_3 = v_1 v_2 = 10$	3	$\dot{v}_3 = v_1 \dot{v}_2 + \dot{v}_1 v_2 = 5$	$\dot{v}_3 = v_1 \dot{v}_2 + \dot{v}_1 v_2 = 2$
4	$v_4 = ln(v_1) = 0.69$	4	$\dot{v}_4 = \frac{1}{v_1} = 0.5$	$\dot{v}_4 = \frac{1}{v_1} = 0$
5	$v_5 = v_3 + v_4 = 10.69$	5	$\dot{v}_5 = \dot{v}_3 + \dot{v}_4 = 5.5$	$\dot{v}_5 = \dot{v}_3 + \dot{v}_4 = 2$
6	$y = v_5 = 10.69$	6	$\dot{y} = \dot{v}_5 = 5.5$	$\dot{y} = \dot{v}_5 = 2$

The forward mode is preferred for $f : \mathbb{R}^n \to \mathbb{R}^m$ when $m \gg n$, as it would require only n runs to calculate the Jacobian matrix. It is generally also more memory efficient, as memory for values that are not needed for future calculations can be freed. However, in situations where $f : R^n \to R$, which is common in machine learning, it would take n evaluations to compute the gradient ∇f.

The reverse mode automatic differentiation is conceptually a bit more evolved. Table 3.2 illustrates how the value and derivatives are calculated. The computations for the value are exactly the same as with forward mode automatic differentiation, but in addition, a computational graph that records the dependencies of elementary functions is created. To calculate the derivatives, the computational graph is used to propagate adjoints $\bar{v}_i$ in reverse. Starting from outputs and traversing the graph until the inputs are reached. Both partial derivatives of the example function are calculated in a single backward pass.

Table 3.2: Example of reverse mode automatic differentiation

↓	Value	↑	$\frac{\partial y}{\partial x_1}$	$\frac{\partial y}{\partial x_2}$
			$\bar{v}_1 = 5.5$	$\bar{v}_2 = 2$
1	$v_1 = x_1 = 2$	6	$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{v_1} = v_2 + \frac{1}{v_1} = 5.5$	
2	$v_2 = x_2 = 5$	5	$\bar{v}_2 = \bar{v}_3 \frac{\partial v_3}{\partial v_2} = v_1 = 2$	
3	$v_3 = v_1 v_2 = 10$	4	$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = 1$	
4	$v_4 = ln(v_1) = 0.69$	3	$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{v_4} = 1$	
5	$v_5 = v_3 + v_4 = 10.69$	2	$\bar{v}_5 = \bar{y} \frac{\partial y}{\partial v_5} = 1$	
6	$y = v_5 = 10.69$	1	$\bar{y} = 1$	

Reverse mode is preferred for $f : R^n \to R^m$ when $n \gg m$. It is also the main mode used in many deep learning applications, because in case of a scalar error $m = 1$, and calculating the full gradient would require only a single backward pass. The downside of reverse mode automatic differentiation is the increased memory requirement. The computational graph and at least some intermediate results from the function evaluation pass need to be stored for the backward pass.

Implementing reverse mode automatic differentiation efficiently is a non-trivial programming task. Fortunately, since it is so essential for training deep neural networks, there are many high quality deep learning libraries implementing the backpropagation algorithm with minimal effort[1]. In this paper I am using PyTorch, which is particularly popular among machine learning researchers in academia[2].

In this paper, I only cover applications of reverse mode automatic differentiation, or backpropagation, in the context of a simple infinite horizon heterogeneous agent model. First, I would like to keep the model as simple as possible to illustrate the benefits of backpropagation. Second, a typical solution algorithm, e.g. value function iteration, can require hundreds of iterations to find a fixed point[3]. Hence, if the backpropagation is feasible in an infinite horizon setting, it should also be for life-cycle models solved using backward induction that typically require fewer iterations.

1. Some of the most popular machine learning libraries include: Tensorflow and JAX by Google, and PyTorch by Facebook. All of them are aimed at being used from Python, although there are interfaces for other languages.

2. Duarte et al. (2020) benchmark different machine learning frameworks and programming languages. They find that Tensorflow is faster than PyTorch. Subjectively, I find the PyTorch to be easier to learn, use, and debug

3. For the infinite horizon problem, using implicit differentiation would be more efficient than the naive application of backpropagation. I do not explore that option in this paper.

3.3 Simple model

In this section I will introduce a standard Bewley-Huggett-Aiyagari type heterogeneous agent incomplete markets model. The purpose of this model is to fix notation and help in illustrating the practical use of backpropagation in conjunction with traditional grid based solution methods: value function iteration and endogeneous grid method.

The economy is populated with a unit mass of *ex-ante* identical infinitely-lived households. Each period a household receives idiosyncratic endowment in a perishable consumption good $y \in Y$. The process for the endowments is governed by a finite state Markov chain, with transition matrix Π and realisations Y[4]. For interest rate r households can borrow and save in a risk free asset a. Household problem in recursive formulation is given in Equation (3.1).

$$\begin{aligned} V(y,a) &= \max_{c,a'} \frac{c^{1-\gamma}-1}{1-\gamma} + \beta \mathbb{E}_{y'|y}[V(y',a')] \\ \text{s.t.} \quad c + a' &= (1+r)a + y \\ a &\geq \underline{a} \end{aligned} \tag{3.1}$$

3.4 Value function iteration as a convolutional neural network

I will first describe how to solve the model with a standard value function iteration (VFI) algorithm, where the choice of assets is restricted to the asset grid. This illustration will serve two purposes. First, it shows what modifications are necessary to the standard algorithm to make it compatible with backpropagation. Second, how the value function iteration can be viewed as a recurrent convolutional neural network.

How can we use VFI together with backpropagation? Fortunately, apart from programming the model in one of the machine learning framework, the value function algorithm itself is standard except for line 7. Conventionally the policy function would be obtained by $g^a(a,y) = \arg\max_{a'} Q^\star(a,a',y)$, which is not differentiable. A workaround is to use a 'soft' version of *argmax* function.

4. The transition probabilities and states of the Markov chain are obtained from discretizing $log(y_{t+1}) = \rho log(y_t) + \varepsilon_t, \varepsilon \sim \mathcal{N}(0,\sigma_\varepsilon)$ using Rouwenhorst method.

Algorithm 1: Value function iteration

Input: β, Π, Reward array $R(a,a',y)$ constructed as in (3.2)
Output: Value array $V^\star(a,y)$, Asset choice array $A(a,a',y)$

Initialize $V^0(a',y')$;
while *error* $>$ *tolerance level* **do**
  $Q^n(a,a',y) = R(a,a',y) + \beta \sum_{y'} \pi(y'|y) V^n(a',y')$;
  $V^{n+1}(a,y) = \max_{a'} Q^n(a,a',y)$;
  $error = ||V^{n+1} - V^n||_\infty$;
end
$A(a,a',y) = \frac{\exp(Q^\star(a,a',y)/\tau)}{\sum_{a'} \exp(Q^\star(a,a',y)/\tau)}$;
begin
  Non-stochastic simulation as in Young (2010)
end

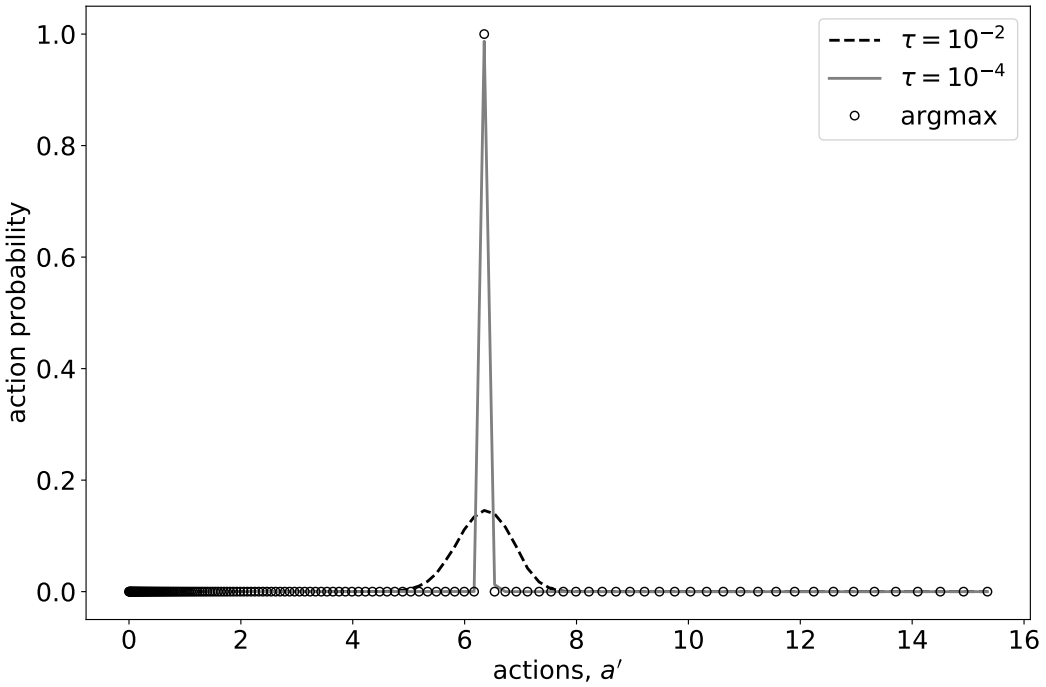
$$R(a,a',y) = \begin{cases} u(c) & \text{if } c = y + (1+r)a - a' > 0 \\ -10^6 & \text{if } c \leq 0 \end{cases} \tag{3.2}$$

This change is equivalent to adding extreme value type I distributed taste shocks into the model, which would result in the same closed form solution for asset choice probabilities in Equation (3.3). Taste shocks can be interpreted as something unobservable for the econometrician affecting the choice of the agent in the model or a logit smoothing (Iskhakov et al. 2017). Here, it is a necessary concession to be able to apply the backpropagation algorithm.

$$\begin{aligned} &\text{Softmax function } \sigma : \mathbb{R}^K \to \mathbb{R}^K \\ &\sigma(\mathbf{z})_i = \frac{\exp(z_i/\tau)}{\sum_{j=1}^K \exp(z_j/\tau)}, \quad for\ i = 1,...,K\ and\ \mathbf{z} = (z_1,...,z_K) \in \mathbb{R}^K \end{aligned} \tag{3.3}$$

The softmax function returns a vector of action probabilities given by the Equation (3.3). With the parameter τ we can govern how concentrated the probabilities are around the action with the highest value, as demonstrated in Figure 3.1. While adding the taste shocks does change the model, we can very closely approximate the original one by choosing a low value for τ.

Figure 3.1 shows the effect of different values of τ on the probabilities of the next period asset choice for a given combination of income y and asset a. As $\tau \to 0_+$ the probability of picking the optimal a' approaches one.



Note: The figure shows the action probabilities of the household with income $y = 7.5320$ and asset $a = 3.7557$.

Figure 3.1: Asset choice probabilities

Using the action probabilities and the transition matrix for the income process, it is easy to find the stationary distribution using a non-stochastic simulation algorithm as in Young (2010). It requires no modifications.

What is the connection between value function iteration and convolutional neural networks? Convolutional neural networks (LeCun et al. 1989), or CNNs, are a type of neural network used for process data structured in a grid (Goodfellow, Bengio, and Courville 2016)[5]. A CNN is comprised of sequence of convolution and max-pooling layers applied to a grid structure (e.g. pixels in a picture).

Convolution operation, as it is implemented in CNNs, slides a window (a convolutional kernel) over the grid structure, calculating weighted average of the grid values in the window, before sliding forward. Similarly, the pooling operation slides a window over the grid structure, calculates the maximum within the window, before moving the to the next set of grid points.

In the value function iteration algorithm the grid like structure is the reward and value array; line 3 can be expressed as a convolution, where the kernel weights are constructed from discounted transition probabilities; line 4 is the same as a pooling over all the values of a'.

5. The paper gives a detailed description of CNNs and the mathematical operations of convolution and pooling they employ.

The CNN implementing the value function iteration algorithm will have as many convolution and max-pooling layer pairs as it takes iterations to converge.

This connection implies that the machine learning frameworks used to implement CNNs have all the necessary ingredients for VFI implementation with minimal effort. CNNs are widely used in image processing and computer vision applications, which often have much higher computing requirements than typical dynamic programming problems solved in economics. This has lead to development of specialized hardware. Most of the relevant software packages have been programmed with distributed and GPU computing in mind, making them one of the easiest ways to use really powerful computing hardware efficiently with very little additional effort. Thus, the connection between the two algorithms could allow computational economists to benefit from the advances in machine learning.

Finally, CNNs weights are trained using advanced gradient-based optimization algorithms. The same algorithms can be used for computational economics applications such as finding equilibrium prices, calculating transition paths, and calibrating or estimating model parameters. Similarly to the advances in hardware, improvements in those optimization algorithms can carry over to economics.

3.5 Endogenous grid method

In the simple model, the asset choice is continuous. The problems with applying VFI could have been avoided by using a solution algorithm that allows the asset choice to lie between grid points[6]. One of the most efficient algorithms for this type of problem is endogeneous grid method (EGM) developed by Carroll (2006).

The implementation of the EGM is standard and requires no modifications for the back-propagation to work[7]. It is especially suitable because it avoids root-finding operations. This is particularly desirable for three reasons: the calculation of the forward trace is very fast; the resulting computational graph is simpler and, thus, requires less memory; there is no need to implement differentiable non-linear solver. To calculate the stationary distribution, I once again use the non-stochastic simulation algorithm of Young (2010).

It is remarkable, that programming the model in one of the machine learning frameworks requires almost no additional effort compared to writing the same program in a vectorized

6. Although, the same strategy used to make VFI compatible with backpropagation could be used in models with discrete choices.

7. I have implemented the algorithm in a vectorized manner because loops tend to be slow in Python and each function call comes in PyTorch comes with a small overhead that would quickly add up.

fashion using more standard numerical libraries. In terms of programming cost, using backpropagation is almost a free lunch.

There is no neural network architecture I know that is similar to endogenous grid method. However, a recent trend in machine learning research is to incorporate more domain-specific knowledge when solving problems in a particular field. This has lead to a 'differentiable programming' paradigm in which computer programs are written in a way that makes it possible to use automatic differentiation, and optimize them with gradient-based methods. As a result, the machine learning libraries have become more like general-purpose scientific computing libraries that allow efficient automatic differentiation.

3.5.1 Computational overhead

What is the additional computational cost of being able to use backpropagation? Theoretically, as described in Section 3.2, it should be around the same as the cost of evaluating the function itself. In practice, it depends on the machine learning framework used. I am using PyTorch (Paszke et al. 2017), which has an overhead of one microsecond per operator execution. To put it in a more meaningful context, Table 3.3 compares the computation times for solving the model and calculating the partial derivatives of aggregate assets with respect to model parameters: β, γ, r, and all the values of Π. All together there are 67 parameters.

Table 3.3: Computational cost

Method	Function		Derivatives	Relative
	With graph (1)	No graph (2)	Backprop. (3)	(3) to (1)
VFI (CPU)	861 $\pm$ 20.3 ms	744 $\pm$ 11.9 ms	1800 $\pm$ 52.8 ms	2.0
VFI (GPU)	190 $\pm$ 2.26 ms	171 $\pm$ 1.89 ms	195 $\pm$ 1.02 ms	1.0
EGM (CPU)	951 $\pm$ 25.6 ms	895 $\pm$ 10.5 ms	782 $\pm$ 5.91 ms	0.8
EGM (GPU)	282 $\pm$ 2.42 ms	240 $\pm$ 0.87 ms	304 $\pm$ 10.5 ms	1.1

Note: The table reports mean $\pm$ one standard deviation time based on 50 function calls. Time is measured in milliseconds (ms). VFI stands for value function iteration and EGM - endogenous grid method. I used 300 asset grid point and 8 income states. The parameters of the model are in appendix 3.A. Computed using AMD Ryzen 5 2600 (CPU) or Nvidia GeForce GTX1080Ti (GPU).

First column shows the time required to solve the model and create a computation graph that will be later used for backpropagation. Second column shows the time required to just solve the model without creating the graph. The program code is otherwise identical. Comparing the first and second column gives a rough estimate of the computational overhead one might expect to see for similar heterogeneous agent models. The additional cost is

between 6-17% of the time it takes to evaluate a function without creating its computation graph.

The benefit of backpropagation comes from being able to calculate the derivatives faster. It takes about the same amount of time to calculate all the derivatives using backpropagation as it takes to solve the model. One sided finite difference method would have required solving the model additionally 67 times. Moreover, the resulting derivatives would have been less accurate because of truncation and rounding errors[8].

3.6 Applications

There are three broad categories of tasks which would benefit from quick and accurate derivatives: comparative statics, clearing markets and calculating transitions, and taking models to the data.

3.6.1 Comparative statics

As the models become more complex it is easy to lose track of what is going on. Being able to automatically differentiate the model would help building the intuition. Although this simple model is well understood, let's suppose we are interested in how the level of aggregate assets depends on the transition probabilities of the income process. This is illustrated on Figure 3.2.

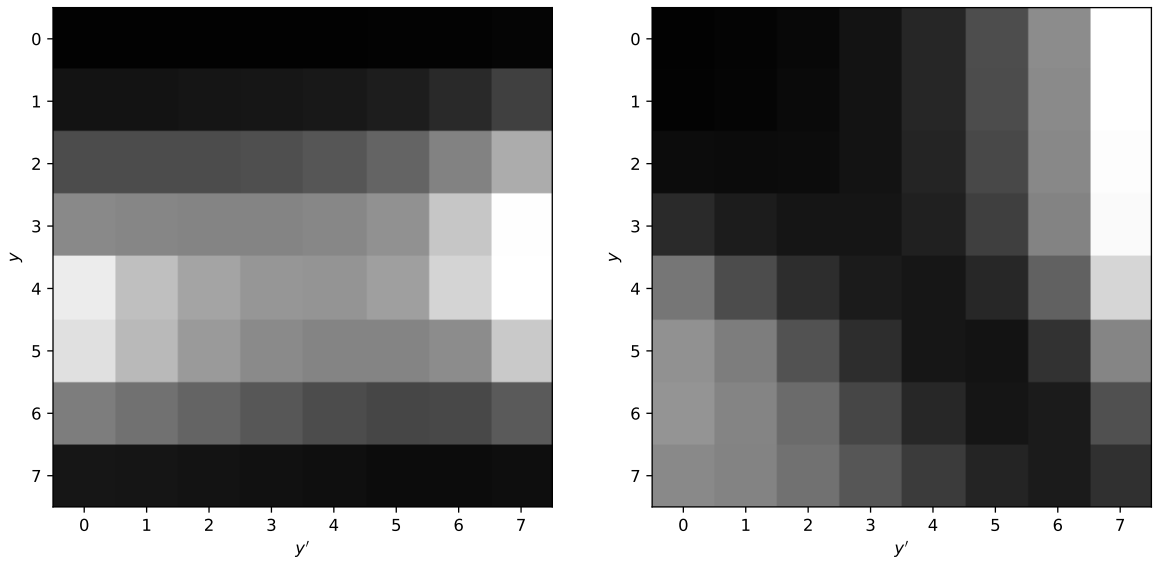
Two things are immediately visible in the above figure. First, aggregate assets are larger when the transition probability from low to high income state is higher. When these probabilities are high, there are more people with high income state, and they can save more. Second, with higher transition probabiility from high to low income state the precautionary motive for saving is stronger. Both of these factors lead to larger aggregate assets.

Here the main benefit of back-propagation is accuracy and speed. To calculate the derivatives using finite differences requires almost a minute, while the backpropagation took less than a second.

3.6.2 Clearing markets and transition dynamics

For clearing the markets and calculating transition paths researchers typically rely on derivative-free methods. Compared to gradient-based optimization algorithms, they usually require more

8. Finite difference method is extremely helpful while debugging and checking for potential implementation errors. All the gradients calculated using backpropagation were close to those obtained with finite difference method.



Note: The left figure illustrates all the derivatives of aggregate assets with respect to income transition probabilities as a heatmap. The right figure normalizes the rows by dividing each row element with the row mean. All the values are positive. Lighter colours refer to larger impact on aggregate assets.

Figure 3.2: How aggregate assets depend on income transition probabilities

function evaluations and converge slower. One solution is to calculate derivatives numerically using finite difference approximation. This can become too expensive to compute when there are many markets or long transition paths.

In the simple model in the Equation (3.1), the interest rate r is assumed to be exogenous. Suppose now that the bonds are in zero net supply. To find the market clearing interest rate, we would first solve the model and calculate the aggregate amount of bonds $\mathscr{A}$ in the economy. Then evaluate the loss function.

$$\Psi(r|\theta) = (\mathscr{A}(r|\theta) - 0)^2 \tag{3.4}$$

In this case, we can use the backpropagation algorithm to find $\frac{\partial \Psi(r|\theta)}{\partial r}$, and update the interest rate using gradient descent algorithm $r^{new} = r - \lambda \frac{\partial \Psi(r|\theta)}{\partial r}$, where λ is the learning rate. Repeating the procedure until convergence.

To calculate the transition dynamics, similarly to conventional algorithm, we would start from guessing an interest rate path $\{r_t\}_{t=0}^{T}$; solve backwards from the final steady state; construct a transition function and iterate the initial distribution forward to calculate a

sequence of aggregate amount of bonds $\{\mathscr{A}_t\}_{t=0}^{T}$. Finally, evaluate the loss function:

$$\Psi(\{r_t\}|\theta) = \frac{1}{T}\sum_{t=0}^{T}(\mathscr{A}_t - 0)^2 \tag{3.5}$$

Using the backpropagation algorithm, we can calculate all the derivatives $\frac{\partial\Psi(\{r_t\}|\theta)}{\partial r_t}$ for $t \in \{0,\dots,T\}$ in a single backward pass. And update our guess for the interest rate path $r_t^{new} = r_t - \lambda\frac{\partial\Psi(\{r_t\}|\theta)}{\partial r_t}$ for $t \in \{0,\dots,T\}$.

Of course, we are not limited to the simple gradient descent algorithm; there exist more sophisticated optimizers that make better use of the gradient information. It is notable that all the necessary derivatives can be evaluated in a single backward pass regardless of the number of markets to clear or transition paths.

3.6.3 Calibration and estimation

Estimating model parameters is computationally expensive, which explains the small number of papers trying to estimate heterogeneous agent models. At the same time, microdata that could be used for estimation has become more abundant and available.

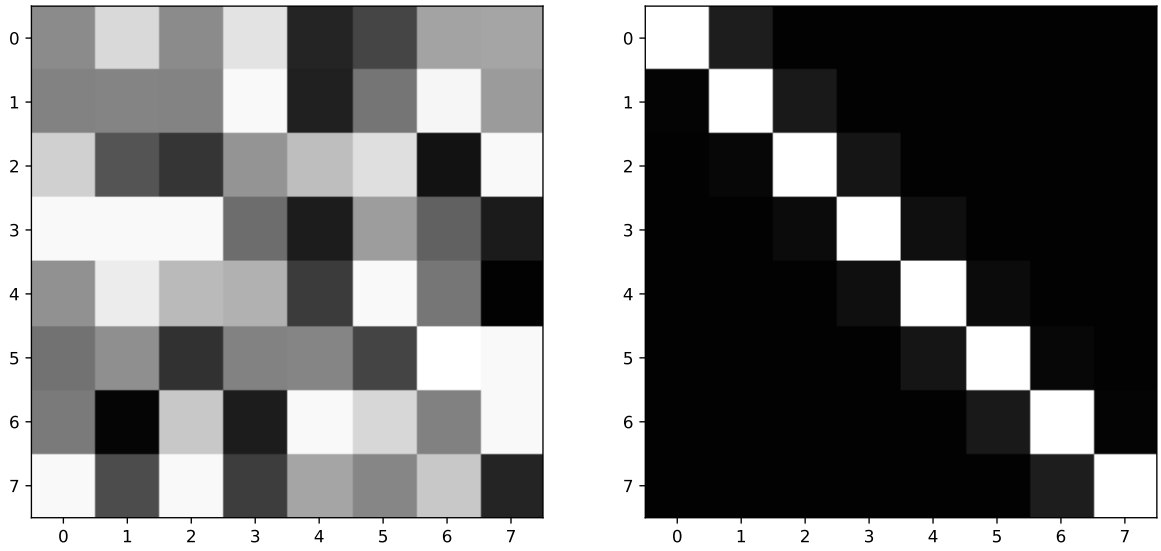
Basic estimation procedure typically involves following steps: (a) initialize parameters, (b) solve the model, (c) calculate moments from the model, (d) compare with data and calculate difference (e) update the parameters to minimize the difference. Where the steps steps (a) to (e) are repeated until minimum is found.

There are two basic ways of speeding up this process: solving the model faster or using a more sophisticated optimizer that converges in fewer iterations. I am already using endogenous grid method, one of the fastest solution algorithms out there. This leaves the second option.

Smarter optimization algorithms typically require more information about the minimization problem. In addition to the function value, many of them can make use of the gradient. However, calculating the gradient using finite differences can be so costly that using derivative-free solver is more efficient. Alternatively, the backpropagation algorithm can calculate the gradients very cheaply.

Using the same simple model I will conduct the following experiment. First, I calculate the stationary distribution of agents. I treat this distribution as if it came from the microdata. Then, I forget the 64 transitions probabilities for the income process, setting them to random values, while making sure they are still valid probabilities and the rows in the transition matrix Π sum to one. Figure 3.3 shows both the true and randomly-initialized transition matrix of

the income process. Third, I solve the model, calculate the stationary distribution, and use the Adagrad algorithm[9] (Zeiler 2012) to estimate the parameters of the transition matrix.



Note: The right plot shows the true transition matrix. The left plot shows the random initialization of the Markov transition matrix of the income process. Lighter values correspond to higher transition probabilities.

Figure 3.3: Starting transition probabilities

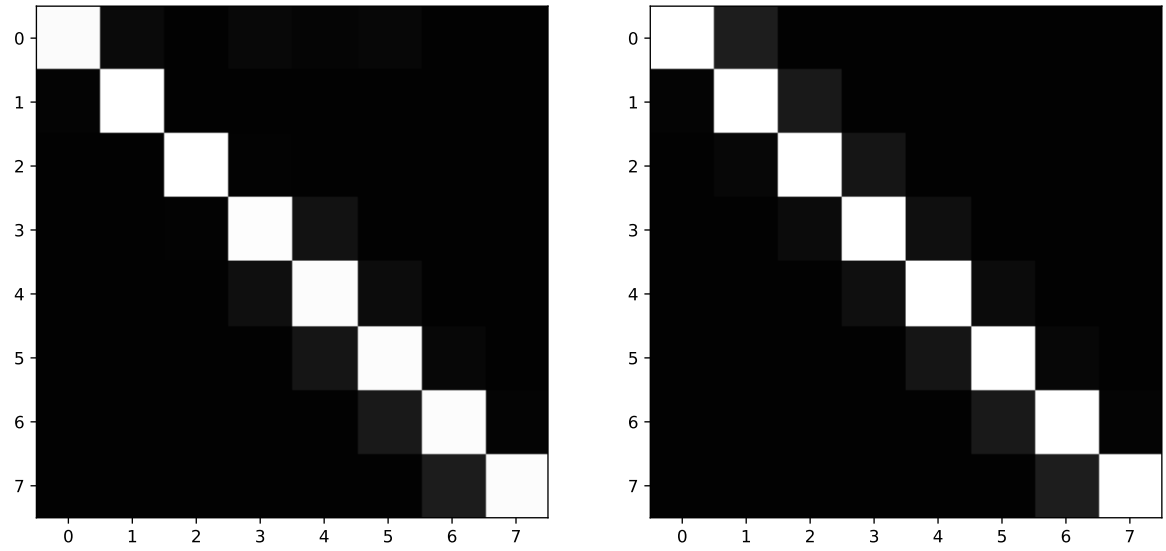
After only about 1000 iterations, the original transition matrix has almost been recovered, as shown in Figure 3.4. This result has signifcant implications. It may be feasible to estimate the transition probabilities in the Markov transition matrix of the income process directly from the microdata on assets and income[10].

3.7 Conclusion

In this paper, I show how the backpropagation algorithm can be used in the context of heterogeneous agent models. I explore three applications: comparative statics, market clearing and transition dynamics, and calibration and estimation. In addition I highlight the connection between the value function iteration algorithm and a recurrent convolutional neural network, a type of deep neural network architecture commonly found in image processing and computer vision applications. Hopefully, the backpropagation algorithm will find it's way into the toolbox of many computational economists.

9. It is a gradient-based optimization algorithm that minimizes the mean squared error between the two distributions.

10. Similar approach for finding the transition probabilities is used by Castañeda, Díaz-Giménez, and Ríos-Rull (2003).



Note: The left plot shows the Markov transition matrix of the income process recovered after 1000 iterations. The right plot shows the true transition matrix. Lighter values correspond to higher transition probabilities.

Figure 3.4: Transition probabilities after 1000 iterations

Appendix 3.A Parameters of the model

Table 3.4: Parameters of the model

Parameter	Description	Value
β	Discount factor	0.97
γ	Risk aversion	1.00
r	Risk-free interest rate	0.005
ρ	Persistence of the income process	0.975
σ_ε	standard deviation of ε	0.92
$\underline{a}$	Borrowing limit	0.0

References

Azinovic, M., L. Gaegauf, and S. Scheidegger. 2019. *Deep Equilibrium Nets* [in en]. SSRN Scholarly Paper ID 3393482. Rochester, NY: Social Science Research Network.

Carroll, C. D. 2006. "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems" [in en]. *Economics Letters* 91 (3): 312–320.

Castañeda, A., J. Díaz-Giménez, and J.-V. Ríos-Rull. 2003. "Accounting for the U.S. Earnings and Wealth Inequality." *Journal of Political Economy* 111 (4): 818–857.

Duarte, V. 2018. *Machine Learning for Continuous-Time Economics* [in en]. SSRN Scholarly Paper ID 3012602. Rochester, NY: Social Science Research Network.

Duarte, V., D. Duarte, J. Fonseca, and A. Montecinos. 2020. "Benchmarking Machine-Learning Software and Hardware for Quantitative Economics" [in en]. *Journal of Economic Dynamics and Control* 111:103796.

Fernández-Villaverde, J., G. Nuño, G. Sorg-Langhans, and M. Vogler. 2020. *Solving High-Dimensional Dynamic Programming Problems Using Deep Learning* [in en]. Technical report.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT press.

Igami, M. 2020. "Artificial Intelligence as Structural Estimation: Deep Blue, Bonanza, and AlphaGo." *The Econometrics Journal* 23 (3): S1–S24.

Iskhakov, F., T. H. Jørgensen, J. Rust, and B. Schjerning. 2017. "The Endogenous Grid Method for Discrete-Continuous Dynamic Choice Models with (or without) Taste Shocks." *Quantitative Economics* 8 (2): 317–365.

Iskhakov, F., J. Rust, and B. Schjerning. 2020. "Machine Learning and Structural Econometrics: Contrasts and Synergies." *The Econometrics Journal* 23 (3): S81–S124.

LeCun, Y., B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. 1989. "Handwritten Digit Recognition with a Back-Propagation Network." *Advances in neural information processing systems* 2.

Maliar, L., S. Maliar, and P. Winant. 2021. "Deep Learning for Solving Dynamic Economic Models." [in en]. *Journal of Monetary Economics.*

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. 2017. "Automatic Differentiation in Pytorch."

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. "Learning Representations by Back-Propagating Errors" [in en]. *Nature* 323 (6088): 533–536.

Tamar, A., Y. Wu, G. Thomas, S. Levine, and P. Abbeel. 2017. "Value Iteration Networks." *arXiv:1602.02867 [cs, stat].*

Villa, A. T., and V. Valaitis. 2019. *Machine Learning Projection Methods for Macro-Finance Models* [in en]. SSRN Scholarly Paper ID 3209934. Rochester, NY: Social Science Research Network.

Young, E. R. 2010. "Solving the Incomplete Markets Model with Aggregate Uncertainty Using the Krusell–Smith Algorithm and Non-Stochastic Simulations" [in en]. *Journal of Economic Dynamics and Control,* Computational Suite of Models with Heterogeneous Agents: Incomplete Markets and Aggregate Uncertainty, 34 (1): 36–41.

Zeiler, M. D. 2012. "Adadelta: An Adaptive Learning Rate Method." *arXiv preprint arXiv:1212.5701.*