Deep equilibrium nets*

Marlon Azinovic, Luca Gaegauf, Simon Scheidegger†

First version: May 24, 2019
This version: November 10, 2020

Abstract

We introduce deep equilibrium nets—a deep learning-based method to compute approximate recursive equilibria of economic models featuring a substantial amount of heterogeneity, significant uncertainty, and occasionally binding constraints. Deep equilibrium nets are neural networks that directly approximate all equilibrium functions and that are trained in an unsupervised fashion to satisfy all equilibrium conditions along simulated paths of the economy. Since the neural network approximates the equilibrium functions directly, simulating the economy is computationally cheap, and training data can be generated at virtually zero cost. We apply our method to solve two distinct annually-calibrated overlapping generations models with high-dimensional state-spaces, aggregate uncertainty, illiquid capital, a one-period bond, occasionally binding constraints, and idiosyncratic risk, demonstrating that deep equilibrium nets can solve rich and economically relevant models accurately.

JEL classification: C61, C63, C68, E21.

Keywords: computational economics, deep learning, deep neural networks, global solution method, life-cycle, occasionally binding constraints, overlapping generations.

*We thank Felix Kübler for his extremely valuable comments and support. Furthermore, we acknowledge helpful suggestions by Rick Evans, Lilia Maliar, Serguei Maliar, Jesus Fernandez-Villaverde, John Rust, Karl Schmedders, Takafumi Usui, Tom Winberry, and participants at seminars at the University of Chicago, at the University of Pennsylvania, Penn State University, University of Lancaster, University of Zurich, EPFL, as well as CEF 2019, the SFI Research Days, PASC 2019, and the Econometric Society world congress 2020. This work was generously supported by grants from the Swiss National Supercomputing Centre (CSCS) under project IDs s885, s995, and the Swiss Platform for Advanced Scientific Computing (PASC) under project ID "Computing equilibria in heterogeneous agent macro models on contemporary HPC platforms". Simon Scheidegger gratefully acknowledges support from the MIT Sloan School of Management and the Cowles Foundation at Yale University.

†Azinović at University of Zurich and Swiss Finance Institute, email: marlon.azinovic@bf.uzh.ch; Gaegauf at University of Zurich, email: luca.gaegauf@bf.uzh.ch; Scheidegger at University of Lausanne, email: simon.scheidegger@unil.ch

1 Introduction

Heterogeneity between types of agents, such as hand-to-mouth and non hand-to-mouth consumers (see, *e.g.*, Bilbiie, 2008; Kaplan et al., 2018; Debortoli and Galí, 2017), financial frictions (see, *e.g.*, Dou et al., 2017; Fernández-Villaverde et al., 2016, and references therein), such as borrowing constraints, and distributional channels (see, *e.g.*, Krueger et al., 2016) are widely recognized as key ingredients for modern macroeconomic models. However, solving economic models with these features in discrete-time remains a challenging, and sometimes prohibitive, task. Substantial computational challenges arise because of a) stochasticity, b) very high-dimensional state-spaces, c) strong non-linearities or kinks in the equilibrium functions, and d) irregular geometries of the ergodic set of states. In the presence of all of these features, the curse of dimensionality (Bellman, 1961), which is caused by the high-dimensional state-space, imposes a substantial roadblock. Computational methods to ameliorate the curse of dimensionality are available for special cases, but mostly rely on the absence of at least some of the other features listed above. A model with financial frictions, for instance, often features occasionally binding constraints, which induce kinks in the equilibrium functions. Even methods that are designed to approximate such functions, such as adaptive sparse grids,[1] fail if the dimensionality of the state-space exceeds ~ 20. While there are methods that can handle a subset of a), b), c), and d), there is a need for tractable computational methods that jointly address all four features.[2]

In this paper, we present a novel algorithm based on unsupervised machine learning that leverages recent advances in deep learning[3] to compute approximate recursive equilibria in discrete-time models with a) stochasticity, b) a large amount of heterogeneity, which results in a high-dimensional state-space, c) financial frictions, which induce kinks in the equilibrium functions, and d) an irregular geometry of the state-space. The method we propose directly approximates the equilibrium functions using a deep neural network trained via a variant of mini-batch stochastic gradient descent on simulated data. We refer to our framework, which approximates all equilibrium functions directly, as

[1] For more details on adaptive sparse grids, see Brumm and Scheidegger (2017).

[2] Adaptive sparse grids, for example, only address challenges a), b), and c), while Gaussian processes (see Scheidegger and Bilionis, 2019) only address a), b), and d).

[3] We refer to Murphy (2012) for a general introduction to machine learning and to Goodfellow et al. (2016) for an introduction to deep learning.

deep equilibrium nets. The key idea of our proposed algorithm is to implement the loss function using the model's first-order conditions directly, thereby circumventing the need to obtain labeled data for supervised learning. The states used to train the deep equilibrium net are sampled by simulating the economy; by doing so, the network learns to approximate the equilibrium only where it matters: on an approximation of the ergodic set. In contrast to most grid-based methods, such as standard value function or policy function iteration with full Cartesian grids or sparse grids, our framework is able to approximate the equilibrium on irregularly shaped geometries. This property is one of the key elements for tackling high-dimensional models, as it avoids wasteful computations in parts of the state space where they are not needed (see, *e.g.*, Renner and Scheidegger, 2018).

The main contributions of this paper are fivefold: first, we introduce a grid-free, generic method based on simulations and deep neural networks to compute global solutions to high-dimensional, and potentially highly non-linear, dynamic stochastic models. Second, the formulation of our methodology does not rely on the invocation of non-linear solvers or optimizers, which allows for a swift generation of a substantial amount of training data. Third, through recent advances in neural network research that alleviate the curse of dimensionality,[4] our method is capable of solving rich models that may be a better description of the real world than those used by practitioners to date. Fourth, we present a hybrid parallelization scheme based on Horovod (see Sergeev and Del Balso, 2018) that allows efficient use of contemporary high-performance computing hardware, which can speed up training by orders of magnitude. Fifth, to illustrate the capabilities of our proposed algorithm, we solve two variations of a classic overlapping generation model from the literature, which we extend by adding borrowing constraints, adjustment costs, multiples assets as well as idiosyncratic health-shocks to the model. In addition, code examples that illustrate our methodology are provided under this link: `https://github.com/sischei/DeepEquilibriumNets`.

More specifically, we solve two annually calibrated life-cycle models with agents that live for 56 periods, borrowing constraints, multiple assets, and aggregate uncertainty. The second model additionally features two types of agents and idiosyncratic health-shocks. We chose this type of model for two

[4]This includes methodological advances (*e.g.*, using rectified linear units (relu) as activation functions to facilitate backpropagation), improvements in the software (*e.g.*, performance-optimized code libraries), and hardware (*e.g.*, graphics and tensor processing units).

reasons: first, this setup serves as a demonstrative example for the introduced method because the dimensionality of the state-space increases with the number of periods the agents live for and, therefore, has a clear economic interpretation; second, because of its economic relevance. Explicitly accounting for an agent's life-cycle is of crucial importance to, for instance, social security (see, *e.g.*, Krueger and Kubler, 2006), climate change (see, Kotlikoff et al., 2020a,b), labor (see, *e.g.*, Gervais et al., 2016), and saving decisions. Saving decisions, for example, are driven by precautionary savings and life-cycle motives. The intergenerational wealth distribution hence affects consumption responses to a financial crisis or monetary policy (see, *e.g.*, Wong, 2016). The rising life expectancy and quickly changing demographics render it a pressing task to understand how age and the age distribution affect the economy.

Computationally, substantial challenges arise when working with life-cycle models in high-dimensional settings. In heterogeneous agent models with incomplete markets and uncertainty, traditional methods like value function iteration or time iteration (see, *e.g.*, Judd, 1998; Ljungqvist and Sargent, 2000), in combination with Cartesian grid-based approximation schemes, quickly become infeasible with increasingly many state variables. The curse of dimensionality (see Bellman, 1961)—that is, the exponential dependence of the overall computational effort on the number of dimensions—can be ameliorated by using Smolyak sparse grids (see Krueger and Kubler, 2004) or adaptive sparse grids (see Brumm and Scheidegger, 2017). However, a new challenge arises from these methods: the value functions or policy functions need to be approximated on a hypercube. Since the ergodic set of the states of the economy is frequently irregularly shaped, it often fills out only a tiny fraction of the hypercube (see also Judd et al., 2011). Thus, approximating the economy on a hypercubic domain can be wasteful and, in some cases, render the computation of solutions to models with irregularly-shaped ergodic sets impossible. Furthermore, the researcher may not always be able to determine a hypercube in which the ergodic set of states of the economy lies. These problems can be addressed by grid-free simulation-based methods, which are suitable for high-dimensions, such as Gaussian processes (see Rasmussen, 2004 for an introduction to Gaussian Processes and Scheidegger and Bilionis, 2019 for an application in economics). Gaussian processes, however, are not capable of dealing with a large amount of input data[5] and hence cannot exploit the fact that simulation-based methods

[5]Standard Gaussian processes become computationally intractable for more than $\sim 10^4$ observa-

can generate input data for machine learning purposes at low cost. In contrast, neural networks turn out to successfully address all of these issues: they are suitable for dealing with very high-dimensional problems,[6] they are a grid-free method, and they excel when there is much data[7] available to learn from.

The algorithm introduced in this paper can leverage these features of neural networks to approximate recursive equilibria in economic models with a high-dimensional state-space. We use the neural network to approximate all equilibrium functions directly. This allows us to simulate the economy at virtually zero cost, which in turn enables us to train the neural network on millions of points, which approximate the ergodic set in a grid-free fashion.

The remainder of the paper is organized as follows: section 2 gives a brief review of the related literature. Section 3 describes the benchmark model, which we use to illustrate our algorithm. Section 4 introduces the deep equilibrium net solution method. In section 5, we showcase its performance in the context of the benchmark model and the economic insights we obtain. Section 6 shows an additional application of deep equilibrium nets to a model featuring two types of agents as well as idiosyncratic risk. Section 7 concludes. In addition, we discuss in appendix D the remaining challenges and possible avenues for future research. In appendix E, we present an application of deep equilibrium nets to a simple model for which an analytical solution is known.

2 Literature review

The methodology we propose in this paper contributes to three particular strands of literature on computing recursive equilibria numerically (see fig. 7 in appendix A for an illustrated summary of the positioning of this paper). First, we propose a global solution method and thereby contribute to the literature focusing on the class of solution algorithms suitable for economic models featuring strong non-linearities, and a large amount of uncertainty. Second, our method is simulation-based and grid-free. Grid-free methods do not rely on constructing a rigid grid and therefore can approximate equilibria more efficiently[8]

tions, as their computational complexity scales as $\mathcal{O}(N^3)$, where N is the number of observations.

[6] See, *e.g.*, Grohs et al., 2018; Jentzen et al., 2018; Becker et al., 2018; Sirignano and Spiliopoulos, 2018 for recent papers showing the potential of neural networks to ameliorate the curse of dimensionality when solving PDEs and computing optimal stopping rules

[7] That is, in the order of $\sim 10^6$ or more observations.

[8] Rather than, for example, interpolating the functions within a hypercube, where only a small fraction of the space that is interpolated may be relevant to the equilibrium, grid-free methods

on irregularly shaped state-spaces. Finally, we contribute to the nascent strand of literature in computational economics that makes use of recent advances in machine learning to compute approximate equilibria in dynamic models. More precisely, we use unsupervised deep learning to approximate the equilibrium functions in annually calibrated life-cycle models directly. To the best of our knowledge, we are the first to do this. In the remainder of this section, we will outline the contributions of our algorithm and its relation to existing research in greater detail.

Since excellent reviews for the existing computational methods are available, our literature review focuses on placing our paper within the emerging strand of literature that makes explicit use of machine learning to compute equilibria in economic models. Maliar and Maliar (2014) and Kollmann et al. (2011) give very detailed overviews of computational methods for economic models with a finite number of agents. For an overview of local and global methods, see Aruoba et al. (2006) and Fernández-Villaverde et al. (2016).

The necessity of global solution methods for models with strong non-linearities is pointed out, for instance, in Dou et al. (2017) and Fernández-Villaverde et al. (2015). Sparse grids (see, *e.g.*, Krueger and Kubler, 2004) and adaptive sparse grids (see, *e.g.*, Brumm et al., 2017; Brumm and Scheidegger, 2017; Scheidegger and Treccani, 2018) can ameliorate the curse of dimensionality but need to approximate the equilibrium on a hyper-cubic domain. See Judd et al. (2011), Maliar and Maliar (2015), and Scheidegger and Bilionis (2019) for the advantages of grid-free, simulation-based methods for models with a high-dimensional state-space and irregularly shaped ergodic sets.

The quest for computational methods, which can jointly address the obstacles of stochasticity, a high-dimensional state-space, strong non-linearities, and irregular state-space geometries, led to the development of a new branch of grid-free methods, which makes explicit use of machine learning to compute equilibria in economic models. The extremely active branch of machine learning research and the related literature from applied mathematics provide a rich set of powerful tools for approximating functions in high-dimensional settings, determining ergodic and feasible sets (see, *e.g.*, Renner and Scheidegger, 2018), and solving optimization problems efficiently—issues with which computational economic modeling is constantly confronted.

Neural networks are particularly promising since they have, to some extent,

approximate the equilibrium functions at states that are reached during simulation.

shown the potential to overcome the curse of dimensionality when solving PDEs and computing optimal stopping rules (Grohs et al., 2018; Jentzen et al., 2018; Becker et al., 2018; Sirignano and Spiliopoulos, 2018). Early applications of neural networks include Hutchinson et al. (1994), who hedge and price derivative securities, Chen and White (1999), who derive approximation rates for a particular type of time-series data, and Norets (2012), who uses neural networks in the context of estimation and discrete-state dynamic programming. Norets (2012) in particular estimates finite-horizon, dynamic discrete choice models and uses a neural network to approximate expected value functions as a function of the economic parameters and state variables. Duffy and McNelis (2001) compare various methods for approximating and solving a stochastic growth model with parameterized expectations, one of which is a shallow neural network.

Closer to this paper, Duarte (2018a,b) presents a solution algorithm for economic models in continuous time, which uses neural networks to approximate the value function. The parameter updates, as implied by supervised learning, are computed efficiently by combining the Hamilton-Jacobi-Bellman equation with Ito's lemma and automatic differentiation. Fernández-Villaverde et al. (2019) use neural networks to forecast aggregate variables in a continuous-time model of financial frictions. Their algorithm builds on the seminal work of Krusell and Smith (1998), but uses neural networks to parameterize the perceived law of motion of aggregate variables, which allows the forecast of the aggregate variables to be non-linear. In Fernández-Villaverde et al. (2019), the expectations of the households depend on a finite set of moments of the cross-sectional distribution of assets as well as an additional endogenous state variable (the expert's net wealth). The algorithm in Fernández-Villaverde et al. (2019) differs from ours along several dimensions: first, the neural network is only used to obtain the perceived law of motion, whereas the neural network in our method approximates all equilibrium functions. Second, the presented algorithm is developed for continuous-time models, while ours is developed for discrete-time. Third, in contrast to Fernández-Villaverde et al. (2019), who, in addition, make use of equation solvers to compute the equilibrium, our algorithm uses simulation and a version of mini-batch stochastic gradient descent exclusively. In discrete-time settings, Villa and Valaitis (2019) use neural networks to approximate expectations to compute equilibria in a variety of settings where high-dimensional state-spaces and the multicollinearity of state variables

pose a computational challenge. Similar to our algorithm, their method iterates between a stochastic simulation phase and a learning phase. In contrast to Villa and Valaitis (2019), we do unsupervised learning. Maliar et al. (2019) provide a general framework for formulating economic models in a way, which is suitable for the application of algorithms from the field of artificial intelligence. These formulations include the maximization of lifetime reward, the minimization of the Euler equation residuals, and the minimization of the Bellman residuals.[9] As one test case, they solve the Krusell and Smith (1998) model with 2000 state variables. Lepetyuk et al. (2019) use the latter framework to compute a global solution to a challenging version of the "Terms of Trade Economic Model" of the Bank of Canada.

3 The economic model

In this section, we consider an overlapping generation (OLG) model with stochastic production very similar to that discussed in Krueger and Kubler (2004).[10] In this model, the dimensionality of the state-space increases linearly with the number of age-groups. Hence a high-dimensional state-space arises naturally and in an economically meaningful way. In order to demonstrate the applicability of our method in a setting with multiple assets, and occasionally binding constraints, we extend the model of Krueger and Kubler (2004) along two lines: first, we include borrowing constraints and adjustment costs on capital, which renders it an illiquid asset; second, we allow agents to trade a liquid one-period bond subject to collateral constraints. In addition to the computational challenges, these extensions, therefore, allow the model to connect to literature highlighting the role of assets with different liquidity on the cross-sectional consumption response to aggregate shocks (see, *e.g.,* Wong, 2016; Kaplan et al., 2018).

[9] The use of Euler equation residuals as a cost function in the independent work of Maliar et al. (2019) is similar to the algorithm presented in this paper.

[10] Note that the method proposed in this paper, while applied solely to OLG models, has a far broader scope: in principal, its generic design allows the computation of recursive equilibria for many other types of recursively formulated models.

3.1 Uncertainty

The OLG model we consider is in discrete time—that is, $t = 0, 1, \ldots$. In each period, one of Z possible discrete exogenous shocks realizes. We assume that the shocks follow a first-order Markov process with finite support $\mathcal{Z} := \{1, \ldots, Z\}$ with transition matrix Π. We denote the aggregate shock realizing in period t by z_t and a history of aggregate shocks up to period t by $z^t = (z_0, \ldots, z_t)$.

3.2 Households

We study an OLG model where agents live for N periods. There is no uncertainty about the lifetime, and there is one representative household per cohort. At each node z^t, a representative household is born. Households only distinguish themselves by their birth-node $z^{t^{\text{birth}}}$. At time $t \in \{t^{\text{birth}}, ..., t^{\text{birth}}+N-1\}$, we identify the household born at t^{birth} by its current age $s = t - t^{\text{birth}} + 1$. For example, the consumption of the household with age s at period t is denoted by $c^s(z^t)$. We omit the explicit dependence on z^t where there is no risk of confusion and write c_t^s. In each period, the agents alive receive a strictly positive labor endowment, which only depends on the age of the agent. The labor endowment of an agent of age s is denoted by l^s. The price of the consumption good is normalized to one. Furthermore, we assume that households supply their labor endowment inelastically for a market wage $w(z^t) = w_t$. At each node z^t, the agents alive maximize their remaining time-separable discounted expected lifetime utility given by

$$\sum_{i=0}^{N-s} \mathrm{E}_t \left[\beta^i u(c_{t+i}^{s+i})\right], \tag{1}$$

where the discount factor $\beta < 1$. Furthermore, we assume that the utility function $u : \mathbb{R}_{++} \to \mathbb{R}$ is smooth, strictly increasing, strictly concave, and that it satisfies the Inada condition $\lim_{c\to 0} u'(c) = \infty$.

There are two ways for households to transfer consumption over time. First, households can save in risky capital, which is subject to adjustment costs, and hence illiquid. The illiquid savings of household s in period t are denoted by $a^s(z^t) = a_t^s$. The savings will become capital in the next period:

$$a_t^s = k_{t+1}^{s+1}, \ \forall t, \ \forall s \in \{1, \ldots, N-1\}, \tag{2}$$

where k_t^s denotes illiquid capital holdings of age-group s in the beginning of period t. Households sell their capital to the firm at market price r_t. The return to capital accrues into the household's illiquid account. The deposit into the illiquid account by household s in period t is given by

$$\Delta_t^s = a_t^s - k_t^s r_t. \tag{3}$$

Capital is illiquid due to the convex adjustment costs given by

$$\frac{\zeta}{2}(\Delta_t^s)^2, \tag{4}$$

where ζ is the intensity of the capital adjustment costs. Furthermore, we impose an exogenous borrowing limit $\underline{a}$, which can be set to zero when borrowing is prohibited:

$$a_t^s \geq \underline{a}. \tag{5}$$

Next to saving in illiquid, risky capital, households can buy a one-period bond at market price p_t. The bond is in zero net-supply. A bond promises a payoff of 1 in the subsequent period. Let b_t^s denote the bond holding of household s in period t and let d_t^s denote the amount of bonds purchased by household s in period t, so that

$$b_{t+1}^{s+1} = d_t^s. \tag{6}$$

The bond is liquid in the sense that there are no costs associated with adjusting the bond holding. Selling the bond is subject to collateral constraints. To model collateral constraints, we follow Kubler and Schmedders (2003) and use

$$\kappa d_t^s + a_t^s \geq 0, \tag{7}$$

were κ is an exogenous constant, and only capital serves as collateral. If a bond-selling household defaults on the payment in the following period, the value of the corresponding amount of capital is transferred to the bond-buying households; otherwise, there is no punishment.[11] Consequently, households only default if the value of the collateral is smaller than the promised payout

[11] To be precise, we assume that in case of default, the value of the pledged collateral, κr_t, is transferred between the liquid accounts.

of the bond. The realized payout of the bond is then given by

$$\min\{\kappa r_t, 1\}. \tag{8}$$

In our numerical experiments below, we chose κ high enough so that the bond is indeed a risk-free asset, and default does not occur in equilibrium. The budget constraint of household s in period t is therefore given by

$$c_t^s + p_t d_t^s + a_t^s + \frac{\zeta}{2}\left(\Delta_t^s\right)^2 = l^s w_t + b_t^s \min\{\kappa r_t, 1\} + r_t k_t^s. \tag{9}$$

Finally, the agents are born and die without any assets—that is, $k_t^1 = b_t^1 = 0$, and $a_t^N = d_t^N = 0$.

3.3 Firms

There is a single representative firm with a Cobb-Douglas production function, where the total factor productivity (TFP) and the depreciation depend on the exogenous shock alone. Each period, after the shock has realized, the firm buys capital and hires labor to maximize its profits, taking prices as given. The production function is given by

$$f(K_t, L_t, z_t) = \eta_t K_t^{\alpha} L_t^{1-\alpha} + K_t(1 - \delta_t), \tag{10}$$

where K_t denotes the aggregate capital bought, L_t denotes the aggregate labor hired, α denotes the capital share in production, η_t denotes the stochastic TFP, and δ_t denotes the stochastic depreciation.

3.4 Markets

The price of capital (r_t) and wages (w_t) as well as the price of the bond (p_t) are determined by market clearing in competitive spot markets for consumption, capital, labor, and bonds.

3.5 Equilibrium

Following Krueger and Kubler (2004), and accounting for our extensions to the model, we now define a competitive equilibrium for our economy:

Definition 1 (competitive equilibrium) *A competitive equilibrium, given initial conditions* $z_0, \{k_0^s\}_{s=1}^{N-1}, \{b_0^s\}_{s=1}^{N-1}$, *is a collection of choices for households* $\{(c_t^s, a_t^s, d_t^s)_{s=1}^N\}_{t=0}^\infty$ *and for the representative firm* $(K_t, L_t)_{t=0}^\infty$ *as well as prices* $(r_t, w_t, p_t)_{t=0}^\infty$, *such that*

1. *Given* $(r_t, w_t)_{t=0}^\infty$, *the choices* $\{(c_t^s, a_t^s, d_t^s)_{s=1}^N\}_{t=0}^\infty$ *maximize* (1), *subject to* (2), (5), (6), (7), *and* (9).
2. *Given* r_t, w_t, *the firm maximizes profits:*

$$(K_t, L_t) \in \underset{K_t, L_t \geq 0}{\arg\max}\, f(K_t, L_t, z_t) - r_t K_t - w_t L_t. \tag{11}$$

3. *All markets clear: For all* t

$$L_t = \sum_{s=1}^N l_t^s, \quad K_t = \sum_{s=1}^N k_t^s, \quad 0 = \sum_{s=1}^N b_t^s. \tag{12}$$

For our choice of the production function, given in equation (10), the first-order conditions of the firm's maximization problem imply that

$$w_t = (1-\alpha)\eta_t K_t^\alpha L_t^{-\alpha}, \tag{13}$$
$$r_t = \alpha \eta_t K_t^{\alpha-1} L_t^{1-\alpha} + (1-\delta_t). \tag{14}$$

For future reference, we write down the household's optimality conditions as well. The Karush-Kuhn-Tucker (KKT) conditions for any given generation of age $s \in 1, ..., N-1$ at node z^t are given by:

$$(1+\zeta\Delta_t^s)\, u'(c_t^s) = \beta \mathrm{E}_t \left[u'(c_{t+1}^{s+1}) r_{t+1} \left(1+\zeta\Delta_{t+1}^{s+1}\right)\right] + \lambda_t^s + \mu_t^s, \tag{15}$$
$$\lambda_t^s \cdot (a_t^s - \underline{a}) = 0, \tag{16}$$
$$a_t^s - \underline{a} \geq 0, \tag{17}$$
$$\lambda_t^s \geq 0, \tag{18}$$
$$p_t u'(c_t^s) = \beta \mathrm{E}_t \left[u'(c_{t+1}^{s+1}) \min\{\kappa r_{t+1}, 1\}\right] + \kappa \mu_t^s, \tag{19}$$
$$\mu_t^s \cdot (a_t^s + \kappa d_t^s) = 0, \tag{20}$$
$$a_t^s + \kappa d_t^s \geq 0, \tag{21}$$
$$\mu_t^s \geq 0. \tag{22}$$

The generation of terminal age N simply consumes everything it has.

4 Approximation of equilibria with deep neural networks

To describe how to solve the OLG model presented in section 3 with deep equilibrium nets, we proceed in two steps. In section 4.1, we define a functional rational expectations equilibrium for the presented economy. Section 4.2 describes how to use deep neural networks to search for an approximate recursive equilibrium by using a projection method (see, *e.g.*, Judd, 1992 and Gaspar and Judd, 1997). In appendix F, we additionally provide a generic parallelization scheme that can speed up our methodology by orders of magnitude. Note that while we present our method in the context of OLG models, it is generic by construction and could be applied to other discrete-time dynamic stochastic models.

4.1 Functional rational expectations equilibrium

Our proposed algorithm aims at approximating a recursive equilibrium: a function mapping the state of the economy to choices and prices, which are consistent with the equilibrium conditions. Following Spear (1988) and Krueger and Kubler (2004), we call such a function a *functional rational expectations equilibrium* (FREE). For our presented economy, the state of the economy can be described by the current exogenous shock and the distribution of assets holdings across age-groups. More precisely, we define a FREE as follows.

Definition 2 (functional rational expectations equilibrium) *A FREE consists of equilibrium functions* $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_a{}^T,\ \boldsymbol{\theta}_\lambda{}^T,\ \boldsymbol{\theta}_d{}^T,\ \boldsymbol{\theta}_\mu{}^T,\ \theta_p\right]^T : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{4(N-1)+1}$*, where* $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{N-1}$ *denotes the capital investment functions,* $\boldsymbol{\theta}_\lambda : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{N-1}$ *denotes the KKT-multiplier functions for the short-selling constraint on capital,* $\boldsymbol{\theta}_d : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{N-1}$ *denotes the bond investment functions,* $\boldsymbol{\theta}_\mu : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{N-1}$ *denotes the KKT-multiplier functions for the collateral constraint, and* $\theta_p : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}$ *denotes the bond price function, such that for all states* $\mathbf{x} := [z,\ \mathbf{k}^T,\ \mathbf{b}^T]^T \in \mathcal{Z} \times \mathbb{R}^{2N}$*, where* $z \in \mathcal{Z}$ *denotes the exogenous shock, and the capital holding* $\mathbf{k} = [k_1,\ \ldots,\ k_N]^T$ *together with the bond holding* $\mathbf{b} = [b_1,\ \ldots,\ b_N]^T$ *denote the endogenous state (i.e., the distribution of capital holdings and bond holdings) with* $k_1 = 0$ *and*

$b_1 = 0$, *we have for all* $i = 1, \ldots, N-1$:[12]

$$(1 + \zeta\Delta(\mathbf{x})_i)\, u'(c(\mathbf{x})_i) = \beta E_z \left[(r(\mathbf{x}_+)(1 + \zeta\Delta(\mathbf{x}_+)_{i+1})\, u'(c(\mathbf{x}_+)_{i+1}) \right] + \theta_\lambda(\mathbf{x})_i + \theta_\mu(\mathbf{x})_i, \tag{23}$$

$$0 = \theta_\lambda(\mathbf{x})_i \theta_a(\mathbf{x})_i, \tag{24}$$

$$0 \leq \theta_a(\mathbf{x})_i, \tag{25}$$

$$0 \leq \theta_\lambda(\mathbf{x})_i, \tag{26}$$

as well as

$$\theta_p(\mathbf{x}) u'(c(\mathbf{x})_i) = \beta E_z \left[(\min\{\kappa r(\mathbf{x}_+), 1\})\, u'(c(\mathbf{x}_+)_{i+1}) \right] + \kappa\theta_\mu(\mathbf{x})_i, \tag{27}$$

$$0 = \theta_\mu(\mathbf{x})_i (\theta_a(\mathbf{x})_i + \kappa\theta_d(\mathbf{x})_i), \tag{28}$$

$$0 \leq \theta_a(\mathbf{x})_i + \kappa\theta_d(\mathbf{x})_i, \tag{29}$$

$$0 \leq \theta_\mu(\mathbf{x})_i \tag{30}$$

and

$$0 = \sum_{i=1}^{N-1} \theta_d(\mathbf{x})_i, \tag{31}$$

where

$$\mathbf{x}_+ = [z_+,\ 0,\ \boldsymbol{\theta}_a(\mathbf{x})^T,\ 0,\ \boldsymbol{\theta}_d(\mathbf{x})^T]^T, \tag{32}$$

where z_+ *denotes the random exogenous shock in the next period, and where*

$$\Delta(\mathbf{x})_i := \theta_a(\mathbf{x})_i - r(\mathbf{x}) x_{1+i}, \tag{33}$$

$$r(\mathbf{x}) = f_K \left(\sum_{i=1}^{N} x_{1+i}, \sum_{i=1}^{N} l^i(x_1), x_1 \right), \tag{34}$$

$$w(\mathbf{x}) = f_L \left(\sum_{i=1}^{N} x_{1+i}, \sum_{i=1}^{N} l^i(x_1), x_1 \right), \tag{35}$$

$$c(\mathbf{x})_i = l^i w(\mathbf{x}) + \min\{\kappa r(\mathbf{x}), 1\} x_{1+i+N} - \theta_p(\mathbf{x})\theta_d(\mathbf{x})_i - \Delta(\mathbf{x})_i - \frac{\zeta}{2} (\Delta(\mathbf{x})_i)^2 \tag{36}$$

Computing an approximate FREE in this model translates to finding an approximation for the $4 \cdot (N-1) + 1$ equilibrium functions such that the above

[12] With a slight abuse of notation, we set $\theta_a(\mathbf{x}_+)_N = \theta_d(\mathbf{x}_+)_N = 0$. More precisely, $\theta_a(\mathbf{x}_+)$ and $\theta_d(\mathbf{x}_+)$ are $N-1$ dimensional vectors. To be formally correct, we would need to distinguish between $i < N-1$ and $i = N-1$ when writing down the equations.

equations are approximately fulfilled for all exogenous shocks and all values the $2N$-dimensional endogenous state takes.

4.2 Algorithm to train deep equilibrium nets

The goal of our solution framework is to approximate the equilibrium functions $\boldsymbol{\theta}$ (see def. 2) with a deep neural network. To do so, we combine four components. Generically, the four components are given by: (i) a suitable class of function approximators; (ii) a loss function measuring the quality of a given approximation at a given state; (iii) an updating mechanism to improve the approximation; and (iv) a sampling method for choosing states for the updating and the evaluation of the approximation quality.

Specific to our algorithm, we chose deep neural networks as a function approximator. The loss function is implemented using the errors in the equilibrium conditions and the neural network parameters are updated using variants of mini-batch gradient descent. To update the parameters of the neural network, as well as to evaluate the quality of our approximation, we sample states from a simulated path of the economy.

This combination of the four components allows us to a) use a large number of states to assess the quality of our approximation, b) sample the states from the approximated ergodic distribution of states in the economy, c) handle irregular geometries of the ergodic set of states, and d) approximate equilibrium functions with kinks and strong non-linearities. We next outline each of the components separately and then combine them into one algorithm.

4.2.1 Function approximator

We use densely connected feedforward neural networks as function approximators because they combine a set of desirable qualities. Neural networks are universal function approximators (Hornik et al., 1989), can resolve distinct local, highly non-linear features, can handle a large amount of high-dimensional input data, and have shown the potential to overcome the curse of dimensionality to some extent when solving, for instance, partial differential equations and computing optimal stopping rules (Grohs et al., 2018; Jentzen et al., 2018; Becker et al., 2018; Sirignano and Spiliopoulos, 2018). For a general introduction to neural networks and deep learning, see Goodfellow et al. (2016).

Given hyper-parameters $\left\{K, \{m_i\}_{i=1}^K, \{\sigma_i(\cdot)\}_{i=1}^K\right\}$ and trainable parameters

$\boldsymbol{\rho}$, a neural network $\mathcal{N}_{\boldsymbol{\rho}}$ encodes the map

$$\mathbf{x} \to \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) = \sigma_K(\mathbf{W}_K ... \sigma_2(\mathbf{W}_2 \sigma_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) ... + \mathbf{b}_K), \tag{37}$$

where $\mathbf{W}_i \in \mathbb{R}^{m_{i+1} \times m_i}$ are matrices often referred to as weight matrices, and $\mathbf{b}_i \in \mathbb{R}^{m_{i+1}}$ are vectors often referred to as bias vectors. The vector $\boldsymbol{\rho}$ denotes the collection of all entries of the weight matrices and the bias vectors. K is referred to as the number of layers of the neural network and m_i as the number of nodes in layer i. The nonlinear functions σ_i are referred to as activation functions and are applied element-wise to each entry of a vector: $\sigma_i(\mathbf{x}) = [\sigma_i(x_1), \dots, \sigma_i(x_{m_{i+1}})]^T$. A densely connected feedforward neural network is hence given by a sequence of matrix-vector multiplications followed by the application of an activation function.

4.2.2 Loss function

The goal of our algorithm is to approximate the equilibrium functions $\boldsymbol{\theta}$ (see def. 2) by a neural network. In this section, we will introduce a loss function: a measure of the quality of our approximation at a given state of the economy.

Let $\boldsymbol{\rho}$ denote the set of trainable parameters of the neural network and let the neural network, given the set of parameters $\boldsymbol{\rho}$, be denoted by $\mathcal{N}_{\boldsymbol{\rho}}$. The neural network maps the state $\mathbf{x}$ into the approximated equilibrium functions

$$\mathcal{N}_{\boldsymbol{\rho}} : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{4(N-1)+1} : \mathbf{x} \to \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}), \tag{38}$$

where $\mathbf{x} = [z, \mathbf{k}^T, \mathbf{b}^T]^T$, and

$$\begin{aligned}
\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}(\mathbf{x}) &= [\hat{\boldsymbol{\theta}}_a(\mathbf{x})^T, \hat{\boldsymbol{\theta}}_\lambda(\mathbf{x})^T, \hat{\boldsymbol{\theta}}_d(\mathbf{x})^T, \hat{\boldsymbol{\theta}}_\mu(\mathbf{x})^T, \hat{\theta}_p(\mathbf{x})]^T \\
&= [\hat{\theta}_a(\mathbf{x})_1, \dots, \hat{\theta}_a(\mathbf{x})_{N-1}, \hat{\theta}_\lambda(\mathbf{x})_1, \dots, \hat{\theta}_\lambda(\mathbf{x})_{N-1}, \\
&\quad \hat{\theta}_d(\mathbf{x})_1, \dots, \hat{\theta}_d(\mathbf{x})_{N-1}, \hat{\theta}_\mu(\mathbf{x})_1, \dots, \hat{\theta}_\mu(\mathbf{x})_{N-1}, \hat{\theta}_p(\mathbf{x})]^T \\
&=: [\hat{a}(\mathbf{x})_1, \dots, \hat{a}(\mathbf{x})_{N-1}, \hat{\lambda}(\mathbf{x})_1, \dots, \hat{\lambda}(\mathbf{x})_{N-1}, \\
&\quad \hat{d}(\mathbf{x})_1, \dots, \hat{d}(\mathbf{x})_{N-1}, \hat{\mu}(\mathbf{x})_1, \dots, \hat{\mu}(\mathbf{x})_{N-1}, \hat{p}(\mathbf{x})]^T.
\end{aligned} \tag{39}$$

Since the true equilibrium functions $\boldsymbol{\theta}$ are defined by the conditions given in definition 2, our aim is to find parameters $\boldsymbol{\rho}$, such that the resulting approximation of the policy functions fulfills equations (23)–(36) for all $i = 1, \dots, N-1$. Given neural network parameters $\boldsymbol{\rho}$ and a state $\mathbf{x}_j$ of the economy, we define

a measure of the quality of the approximation by evaluating the errors in the equilibrium conditions, which result from the policy functions encoded by the neural network.

For readability, we will define the error in each of the equations separately before summing them to form a single measure encoding all equilibrium conditions.

Firm optimization Some of the equilibrium conditions can be enforced to hold exactly for all states. For any state $\mathbf{x}$ and neural network parameters $\boldsymbol{\rho}$, the equilibrium prices for capital and labor depend on the state alone. We can determine the correct equilibrium prices for capital and labor using equations (34) and (35) and, therefore, satisfy them exactly.

Budget constraint of the household We can also ensure that the budget constraints of the households hold exactly by substituting $\hat{\mathbf{a}}$, $\hat{\mathbf{d}}$, and $\hat{p}$, which are encoded by neural network parameters $\boldsymbol{\rho}$, into the equilibrium policies in equation (36):

$$\begin{aligned}\hat{c}(\mathbf{x})_i = l^i w(\mathbf{x}) + \min\{\kappa r(\mathbf{x}), 1\} x_{1+i+N} - \hat{p}(\mathbf{x})\hat{d}(\mathbf{x})_i \\ - \hat{\Delta}(\mathbf{x})_i - \frac{\zeta}{2}\left(\hat{\Delta}(\mathbf{x})_i\right)^2,\end{aligned} \tag{40}$$

where

$$\hat{\Delta}(\mathbf{x})_i := \hat{a}(\mathbf{x})_i - r(\mathbf{x}) x_{1+i}. \tag{41}$$

State transition The transition from one state to the next, equation (32), can also be satisfied exactly by setting the next period's endogenous state consistent with the current period's policies:

$$\mathbf{x}_+ = [z_+,\ 0,\ \hat{\mathbf{a}}(\mathbf{x})^T,\ 0,\ \hat{\mathbf{d}}(\mathbf{x})^T]^T. \tag{42}$$

Inequalities There are four inequalities (eq. (25), (26), (29), and (30)) from the remaining equilibrium conditions which can be enforced through the architecture of the neural network.

Specifically, using an activation function that has a non-negative range in the output layer, such as the softplus function, can ensure that the inequalities are always satisfied. To allow for this, we redefine the neural network's output such that it approximates $\boldsymbol{\theta}_{\text{col}}(\mathbf{x}) := \boldsymbol{\theta}_a(\mathbf{x}) + \kappa \boldsymbol{\theta}_d(\mathbf{x})$, where $\boldsymbol{\theta}_{\text{col}}(\mathbf{x})$ is the collateral

requirement policy, instead of approximating $\boldsymbol{\theta}_d(\mathbf{x})$ directly. That is,

$$\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) = [\hat{\boldsymbol{\theta}}_a(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\lambda(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_{\text{col}}(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\mu(\mathbf{x})^T,\ \hat{\theta}_p(\mathbf{x})]^T.$$[13]

Household's optimization The equilibrium conditions related to the household's optimization problem are equations (23), (24), (27), and (28). As opposed to the previously discussed conditions, we cannot easily enforce them for all possible states. Therefore, taking the enforced equilibrium conditions, the neural network parameters $\boldsymbol{\rho}$, and a given state $\mathbf{x}_j$ as given, we construct a measure for how well the current approximation fulfills each of the remaining conditions. The error in the Euler equation for capital for generation i, namely equation (23), is given by

$$\begin{aligned} e^{i,\text{EE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho}) =& (1+\zeta\hat{\Delta}(\mathbf{x}_j)_i)u'(\hat{c}(\mathbf{x}_j)_i) \\ &- \beta \mathrm{E}_{z_j}\left[r(\mathbf{x}_{j,+})(1+\zeta\hat{\Delta}(\mathbf{x}_{j,+})_{i+1})u'(\hat{c}(\mathbf{x}_{j,+})_{i+1})\right] \\ &- \hat{\lambda}(\mathbf{x}_j)_i - \hat{\mu}(\mathbf{x}_j)_i. \end{aligned} \tag{43}$$

Following Judd (1992), we convert the Euler equation error of generation i in state $\mathbf{x}_j$, $e^{i,\text{EE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho})$, to the relative Euler equation error defined as

$$e^{i,\text{REE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho}) := \frac{u'^{-1}\left(\frac{\beta \mathrm{E}_{z_j}\left[r(\mathbf{x}_{j,+})(1+\zeta\hat{\Delta}(\mathbf{x}_{j,+})_{i+1})u'(\hat{c}(\mathbf{x}_{j,+})_{i+1})\right]+\hat{\lambda}(\mathbf{x}_j)_i+\hat{\mu}(\mathbf{x}_j)_i}{1+\zeta\hat{\Delta}(\mathbf{x}_j)_i}\right)}{\hat{c}(\mathbf{x}_j)_i} - 1. \tag{44}$$

The advantage of using the relative Euler equation error is that its value has an economic interpretation that is independent of the utility function; namely, it quantifies the consumption error. Note that

$$e^{i,\text{EE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho}) = 0 \Leftrightarrow e^{i,\text{REE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho}) = 0.$$

[13] The approximation of the bond policy is easily backed out from the neural network by $\hat{\boldsymbol{\theta}}_d(\mathbf{x}) = (\hat{\boldsymbol{\theta}}_{\text{col}}(\mathbf{x}) - \hat{\boldsymbol{\theta}}_a(\mathbf{x}))/\kappa$.

Similarly, we define the relative Euler equation error for bond investment (corresponding to eq. (27)) as

$$e^{i,\text{REE bond}}_{\mathbf{x}_j}(\boldsymbol{\rho}) := \frac{u'^{-1}\left(\frac{\beta \mathrm{E}_{z_j}[\min\{\kappa r(\mathbf{x}_+),1\}u'(\hat{c}(\mathbf{x}_{j,+})_{i+1}]+\kappa\hat{\mu}(\mathbf{x}_j)_i}{\hat{p}(\mathbf{x}_j)}\right)}{\hat{c}(\mathbf{x}_j)_i} - 1. \tag{45}$$

The errors in the remaining KKT conditions (eq. (24) and (28)) are given by

$$e^{i,\text{KKT cap}}_{\mathbf{x}_j}(\rho) := \hat{\lambda}(\mathbf{x}_j)_i \hat{a}(\mathbf{x}_j)_i, \text{ and} \tag{46}$$

$$e^{i,\text{KKT bond}}_{\mathbf{x}_j}(\rho) := \hat{\mu}(\mathbf{x}_j)_i(\hat{a}(\mathbf{x}_j)_i + \kappa\hat{d}(\mathbf{x}_j)_i). \tag{47}$$

Market clearing The final equilibrium condition, equation (31), is the market clearing condition of the bond. We define the error in the market clearing condition as

$$e^{\text{mc}}_{\mathbf{x}_j}(\rho) = \sum_{i=1}^{N-1} \hat{d}(\mathbf{x}_j)_i. \tag{48}$$

Loss function encoding the equilibrium conditions If the neural network $\mathcal{N}_{\boldsymbol{\rho}}$ would encode the true equilibrium functions exactly, equations (44)–(48) would evaluate to zero for all states $\mathbf{x}$ of the economy. Therefore, we can define a loss function, *i.e.*, a measure of the quality of the approximation $\mathcal{N}_{\boldsymbol{\rho}}$, by quantifying the extent to which the equations (44)–(48) differ from zero when evaluated at a given state $\mathbf{x}$. An unsupervised loss function generally depends on two components: the parameters of the function approximator $\boldsymbol{\rho}$ and the set of states $\mathbf{x}$ at which the desired conditions are evaluated. The latter is referred to as the *training set*, which we denote as $\mathcal{D}_{\text{train}}$. Given parameters $\boldsymbol{\rho}$ and a set of states $\mathcal{D}_{\text{train}}$, we define the loss function as the mean squared error of all equilibrium conditions:

$$\begin{aligned}\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|}\sum_{\mathbf{x}_j\in\mathcal{D}_{\text{train}}}\Bigg(&\frac{1}{N-1}\sum_{i=1}^{N-1}\left(e^{i,\text{REE cap}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right)^2 + \left(e^{i,\text{REE bond}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right)^2 \\ &+ \left(e^{i,\text{KKT cap}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right)^2 + \left(e^{i,\text{KKT bond}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right)^2\Bigg) \\ &+ \left(e^{\text{mc}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right)^2.\end{aligned} \tag{49}$$

4.2.3 Updating mechanism

This section describes how to use the loss function to optimize the trainable parameters $\boldsymbol{\rho}$.[14] The loss function is defined such that smaller values correspond to lower mean squared errors in the equilibrium conditions. Consequently, parameters are deemed "good" if they minimize the loss function. Due to the functional structure of deep neural networks, variants of gradient descent, are typically used to optimize the parameters $\boldsymbol{\rho}$.[15] Gradient descent updates the parameters step-wise in the direction in which the loss function decreases—that is:

$$\rho_k^{\text{new}} = \rho_k^{\text{old}} - \alpha^{\text{learn}} \frac{\partial \ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}^{\text{old}})}{\partial \rho_k^{\text{old}}} \; \forall k \in \{1, \ldots, length(\boldsymbol{\rho})\}. \tag{50}$$

The parameter $\alpha^{\text{learn}} > 0$ that governs by how much the parameters are adjusted with each gradient descent step is referred to as the *learning rate*. Popular variants of gradient descent that can speed up the learning process are briefly discussed in appendix C.1. Deep and wide neural networks that are trained on limited training data $\mathcal{D}_{\text{train}}$ tend to overfit. That is, the neural network optimizes the parameters $\boldsymbol{\rho}$ on the training set $\mathcal{D}_{\text{train}}$ to the extent that the approximation quality deteriorates for new, unseen states, not part of the training set. Hence, the availability of training data is often a bottleneck in deep learning applications. The formulation of our loss function together with an efficient algorithm to generate new and large training sets $\mathcal{D}_{\text{train}}$ (as introduced in the next section) enables our algorithm to circumvent this bottleneck.

4.2.4 Sampling

As described in the previous section 4.2.3, the parameters of the neural network $\boldsymbol{\rho}$ are chosen to minimize the loss function (eq. (49)). In this section, we describe the training set ($\mathcal{D}_{\text{train}}$) generation process used to obtain a good approximation of the equilibrium functions for the ergodic set of states of the economy.

Since we want to use the approximated equilibrium functions to simulate the modeled economy, they must provide a good approximation for those states, which will be visited during the simulation. Therefore, we chose to train

[14] The hyper-parameters have to be chosen by the modeler prior to the training we describe in this section. The modeler may use prior experience, manual, random, or grid search as well as methods such as Bayesian optimization (see, *e.g.*, Bergstra et al., 2011).

[15] We use Adam (see Kingma and Ba, 2014) together with mini-batch gradient descent. See appendix C.1 for details.

the neural network on the states visited on the simulated path of the economy. To do so, we start with an arbitrary, economically feasible starting state $\mathbf{x}_1^0$, and randomly initialized neural network parameters $\boldsymbol{\rho}$. Then, we simulate $T-1$ periods forward based on the approximated equilibrium functions given by the neural network.[16] Since our method approximates the equilibrium functions directly, simulating the evolution of the economy is computationally cheap. The resulting T simulated states of the economy constitute our dataset $\mathcal{D}_{\text{train}}^0 = \{\mathbf{x}_1^0 \ldots, \mathbf{x}_T^0\}$. We call the set of T simulated periods $\mathcal{D}_{\text{train}}$ an *episode*. We split this input data into mini-batches—smaller subsets of size m with random membership—and perform gradient descent steps on each subset as given in equation (50). A completion of an *epoch* is defined as when the whole dataset $\mathcal{D}_{\text{train}}$ is passed through the algorithm. Per epoch, the neural network parameters are updated T/m times. Next, we set $\mathbf{x}_1^1 = \mathbf{x}_T^0$ and use the updated parameters of the neural network to simulate $T-1$ periods forward, generate a new training set $\mathcal{D}_{\text{train}}^1 = \{\mathbf{x}_1^1, \ldots, \mathbf{x}_T^1\}$, and repeat the process. Since we can generate a large amount of training data in this setting, over-fitting is not a primary concern.[17] As the neural network learns better parameter values, the simulated states become better representatives of the ergodic set of states of the economy.

Since the loss function merely requires a set of feasible states of the economy to be evaluated, our algorithm does not require us to obtain the training data $\mathcal{D}_{\text{train}}$ from simulations. If the ergodic set of states of the economy is known, if one wants to approximate the equilibrium functions on a larger set or if one wants to improve the approximation quality at specific areas of the state-space, the training set $\mathcal{D}_{\text{train}}$ can be chosen accordingly.

4.2.5 Algorithm

In this section, we summarize how the components described in the previous sections are combined to compute approximate recursive equilibria with deep equilibrium nets.

Starting from a randomly initialized neural network, our algorithm iterates between generating a new training set $\mathcal{D}_{\text{train}}$ by simulating a desired amount

[16] Since the neural network parameters are initialized randomly, some corrections have to be made for the case when the neural network predicts an infeasible endogenous state in the next period (such as negative aggregate capital). The adjustments are easy to make and are described in appendix D.3.

[17] Our method allows us to train the neural network on more than a billion simulated states.

of states and improving the parameters $\boldsymbol{\rho}$ of the neural network by performing a variant of gradient descent steps on the training set. The error on a new set of simulated states is an out-of-sample error and can be used to judge the out-of-sample quality of the approximation. Because we use the neural network to approximate the equilibrium functions directly, the simulation of a new set of points is computationally cheap. Therefore, we can set the number of epochs to train on each set of simulated points to one. Consequently, each simulated state is only used for a single gradient descent step, which guards our algorithm against overfitting. Algorithm 1 provides the pseudo-code of the deep equilibrium net.[18] Our parallelization scheme, which can speed up the computations by orders of magnitude, is outlined in appendix F.

Algorithm 1: Algorithm for training deep equilibrium nets.

Data:
T (length of an episode),
N^{epochs} (number of epochs on each episode),
$\tau^{\max}$ (desired threshold for max error),
τ^{mean} (desired threshold for mean error),
$\epsilon^{\text{mean}} = \infty$ (starting value for current mean error),
$\epsilon^{\max} = \infty$ (starting value for current max error),
N^{iter} (maximum number of iterations),
$\boldsymbol{\rho}^0$ (initial parameters of the neural network),
$\mathbf{x}_1^0$ (initial state to start simulations from),
$i = 0$ (set iteration counter),
α^{learn} (learning rate)
Result:
success (boolean if thresholds were reached)
$\boldsymbol{\rho}^{\text{final}}$ (final neural network parameters)
while $((i < N^{iter}) \wedge ((\epsilon^{mean} \geq \tau^{mean}) \vee (\epsilon^{max} \geq \tau^{max})))$ **do**
  $\mathcal{D}^i_{\text{train}} \leftarrow \{\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_T^i\}$ (generate new training data by simulating an episode of T periods as implied by the parameters $\boldsymbol{\rho}^i$)
  $\mathbf{x}_0^{i+1} \leftarrow \mathbf{x}_T^i$ (set new starting point)
  $\epsilon_{\max} \leftarrow \max\left\{\max_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e^{\cdot\cdot}_{\mathbf{x}}(\boldsymbol{\rho})|\right\}$ (calculate max error on new data)
  $\epsilon_{\text{mean}} \leftarrow \max\left\{\frac{1}{T}\sum_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e^{\cdot\cdot}_{\mathbf{x}}(\boldsymbol{\rho})|\right\}$ (calculate mean error on new data)
  for $j \in [1, ..., N^{epochs}]$ **do**
    (learn N^{epochs} on data)
    for $k \in [1, ..., length(\boldsymbol{\rho})]$ **do**

$$\rho_k^{i+1} = \rho_k^i - \alpha^{\text{learn}} \frac{\partial \ell_{\mathcal{D}^i_{\text{train}}}(\boldsymbol{\rho}^i)}{\partial \rho_k^i} \tag{51}$$

      (do a gradient descent step to update the network parameters)
    end
  end
  $i \leftarrow i + 1$ (update episode counter)
end
if $i = N^{iter}$ **then return** (success $\leftarrow False, \boldsymbol{\rho}^{\text{final}} \leftarrow \boldsymbol{\rho}^i$) ;
else return (success $\leftarrow True, \boldsymbol{\rho}^{\text{final}} \leftarrow \boldsymbol{\rho}^i$) ;

[18] For simplicity, the pseudo-code implements gradient descent rather than mini-batch gradient descent.

Discount factor β	Relative risk aversion γ	Capital share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1} = 1.022 \mid \eta_t = 1.022)$ $P(\eta_{t+1} = 0.978 \mid \eta_t = 0.978)$	Persistence depreciation $P(\delta_{t+1} = 0.08 \mid \delta_t = 0.08)$ $P(\delta_{t+1} = 0.11 \mid \delta_t = 0.11)$	Borrowing constraint $\underline{a}$	Adjustment cost capital ζ	Collateral requirement bond κ
0.95	2	0.3	$\{0.978, 1.022\}$	$\{0.08, 0.11\}$	0.905 0.905	0.972 0.700	0.0	0.5	1.1236

Table 1: Parameterization of the benchmark model (*cf.* sec. 3).

5 Deep equilibrium nets in action: a first benchmark model

To evaluate the performance of our algorithm, we study an annually calibrated OLG model, as outlined in section 3. We chose this model because it features the computational challenges we want to address with our solution framework. We model $N = 56$ age-groups—that is, the endogenous state is 110-dimensional.[19] Furthermore, there are aggregate shocks, and occasionally binding constraints. In consequence, the OLG model specification used here offers a unique setting to illustrate the performance of our new solution method for an economically relevant class of models.

First, we detail the parameterization of the model and the hyper-parameters of the neural network. We then demonstrate that our proposed algorithm can solve the model with high accuracy. Lastly, we study the aggregate and cross-sectional consumption response to aggregate shocks.

5.1 Parameterization

In this section, we briefly outline the model parameters (*cf.* sec. 5.1.1) and neural network hyper-parameter specification (*cf.* sec. 5.1.2) that we use in our numerical experiments below.

5.1.1 Model parameters

We study an OLG model when borrowing in capital is prohibited entirely, *i.e.*, $\underline{a} = 0$.

The stochastic depreciation in our model aims to represent rare disasters in a parsimonious and stylized way. In the spirit of Gourio (2012), we chose a depreciation of $\delta = 0.08$ during normal times and $\delta = 0.11$ in the disaster state.

[19] To be precise, a sufficient endogenous state is 108-dimensional since agents enter the economy without any assets.

The probability to enter a disaster during normal times is given by 0.028 and the persistence of the disaster state is chosen to be 0.70, resulting in an average disaster length of roughly 3.33 years. The associated transition matrix is given by

$$\Pi^{\delta} = \begin{bmatrix} 0.972 & 0.028 \\ 0.300 & 0.700 \end{bmatrix}, \tag{52}$$

which is in line with values, which are used in the literature on rare disasters and capital depreciation (see, *e.g.*, Gourio, 2012; Usui, 2019).

The TFP process is modeled as an independent AR(1) process, with an unconditional average of $\bar{\eta} = 1$, a yearly persistence of $\rho_{\eta} = 0.81$, and normally distributed yearly innovations with mean zero and standard deviation of $\sigma_{\eta} = 0.013$. These values are within the standard range of the real business cycle literature (see, *e.g.*, Fernandez-Villaverde and Guerron-Quintana, 2020). We discretize the AR(1) process into a two-state Markov process following Rouwenhorst (1995). More precisely, TFP η takes one of two values, $\eta \in \{0.978, 1.022\}$ and evolves according to the transition matrix

$$\Pi^{\eta} = \begin{bmatrix} 0.905 & 0.095 \\ 0.095 & 0.905 \end{bmatrix}. \tag{53}$$

Depreciation and TFP evolve independently of each other. In total, there are four possible combinations of depreciation shock and TFP shock that are labeled by $z \in \mathcal{Z} = \{1, 2, 3, 4\}$ and correspond to $\delta = [0.08, 0.08, 0.11, 0.11]^T$ and $\eta = [0.978, 1.022, 0.978, 1.022]^T$. The overall transition matrix is given by $\Pi = \Pi^{\delta} \otimes \Pi^{\eta}$, where $\Pi_{i,j}$ denotes the probability of a transition from the exogenous shock i to exogenous shock j, where $i, j \in \mathcal{Z}$.

The labor endowment over the life-cycle is shown in figure 8 in appendix A. Agents are "born" with age 25, work until age 63, and are retired from ages 64 to 80. During the agents' working age, the labor endowment is hump shaped and increases by 127% between age 25 to its peak value at age 53. The shape and rise of the labor endowment roughly mimics the average earnings by age-group documented by Guvenen et al. (2015). Following Brumm et al. (2017), we model the retirement period by a linearly decreasing segment, followed by a constant segment. That is, the labor endowment drops by 50% between ages 64 to 70, whereafter it stays constant.

Following Krueger and Kubler (2004), the capital share of production is set to $\alpha = 0.3$, patience is set to $\beta = 0.95$, and the agents' utility from consump-

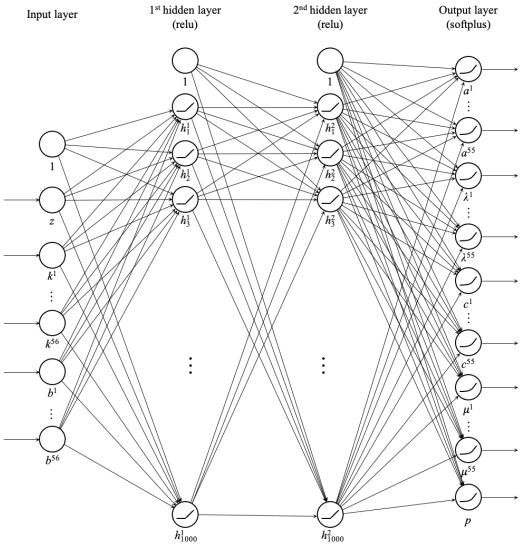


Figure 1: Schematic illustration of the neural network architecture for the deep equilibrium net. For simplicity, it is illustrated for the case where no redundant information is passed to the neural network. See appendix D for explanations on providing additional inputs to the neural network.

tion exhibit constant relative risk aversion of $\gamma = 2$. The parameter κ, which determines the collateral requirement, is set, such that agents can borrow at most as much as is covered by the minimum value of their depreciated capital in the next period: $\kappa = \frac{1}{1-0.11} \approx 1.12$. The bond is hence a risk-free asset. The parameter ζ, governing the adjustment costs of capital, is set to 0.5. A summary of our parameterization is given in table 1.

5.1.2 Neural network hyper-parameters

We use a deep neural network with two hidden layers to solve our benchmark model. Heuristically, we found it to be helpful to augment the state of the economy with redundant information before passing it to the neural network.[20] The input layer consists of $12 + 4 \cdot N$ input nodes, which for $N = 56$ results in 236 input nodes. After the input layer, the neural network features two hidden layers with 1000 relu-activated hidden nodes each. The output layer consists of $4 \cdot (N - 1) + 1 = 221$ nodes, activated with softplus functions to ensure that the non-negativity constraints are fulfilled. A schematic illustration of the neural network architecture, for clarity displayed without providing redundant

[20] See appendix D.2 for more details.

Episodes	Learning rate α^{learn}	Periods per episode T	Epochs per episode N^{epochs}	Mini-batch size m	Nodes hidden layers	Activations hidden layers
1 – 60,000	1×10^{-5}	10,000	1	64	1,000 1,000	relu relu
60,000 – 200,000	1×10^{-6}	10,000	1	1,000	1,000 1,000	relu relu

Table 2: Hyper-parameters chosen to train the deep equilibrium net for the benchmark model.

information, is given in figure 1.[21] We keep the number of epochs per episode N^{epochs} small to avoid overfitting. The learning rate has to be chosen small enough, and the mini-batch size large enough, so that the mini-batch gradient descent steps are not too noisy. Heuristically, we found it helpful to decrease the learning rate and increase the mini-batch size towards the end of the training procedure in order to fine-tune the neural network parameters. Therefore, we switch from mini-batch size $m = 64$ and learning rate of $\alpha^{\text{learn}} = 10^{-5}$ to a larger mini-batch size $m = 1000$ and a smaller learning rate $\alpha^{\text{learn}} = 10^{-6}$ after $60,000$ episodes of training.[22] The chosen hyper-parameters are specified in table 2. To accelerate the simulation, we simulate 1000 ten-period paths in parallel, rather than a single 10,000-period path.

5.2 Training progress of the deep equilibrium net

Next, we illustrate the performance of the proposed algorithm for our benchmark model. Figure 2 shows the evolution of the cost function during the first 200,000 episodes of training: the cost function decreases quickly during training and converges below $10^{-7.0}$. The step-like decrease after $60,000$ episodes stems from the increase in the mini-batch size and the decrease in the learning rate, as described in the previous section.[23] Figure 3 shows the distribution of investments in capital and the bond (first row), consumption (second row), and relative Euler equation errors (third row), as well as error in the bond market clearing conditions (fourth row) on a simulated path of 2000 simulated states after 1, 1000, and 200,000 episodes of training. As the training process advances,

[21] As typical when working neural networks, the architecture presented here is not the result of theoretically motivated performance optimization, but based on numerical experimentation.

[22] Increasing the batch-size has the effect that the gradient estimates are less noisy. Decreasing the learning rate results in smaller steps being taken. The initial hyper-parameters are higher to accelerate the initial rough-grain stage of training.

[23] The training of the benchmark model typically consumes less than 10 minutes of compute time on a single hybrid CPU/GPU node to reach mean relative Euler equation errors below 1%.

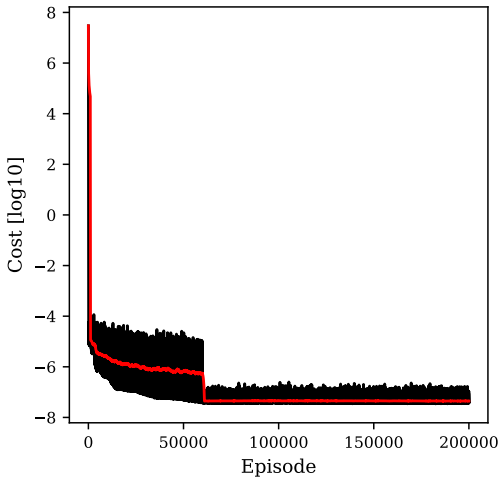


Figure 2: Evolution of the cost function during 200,000 episodes of training, 1 epoch per episode. The black line shows the cost function evaluated on a newly simulated episode, the red line shows a moving average over the last 1000 episodes. The corresponding neural network parameters are described in section 5.1.

	mean	max	0.1	10	50	90	99.9
Rel Ee capital [%]	0.024	0.528	0.000	0.002	0.013	0.060	0.324
Rel Ee bond [%]	0.021	0.608	0.000	0.002	0.012	0.051	0.255
KKT capital [$\times 10^{-2}$]	0.000	0.127	0.000	0.000	0.000	0.000	0.000
KKT bond [$\times 10^{-2}$]	0.005	0.104	0.000	0.000	0.000	0.000	0.021
Market clearing [$\times 10^{-2}$]	0.053	0.333	0.000	0.017	0.047	0.089	0.252
Market clearing (normalized) [$\times 10^{-2}$]	0.000	0.002	0.000	0.000	0.000	0.000	0.001

Table 3: The first two rows show the relative Euler equation errors (Rel Ee in %) on 10,000 simulated periods. Rows 3 and 4 show the errors in the KKT conditions ($\times 10^{-2}$). Rows 5 and 6 show the errors in the market clearing conditions ($\times 10^{-2}$). In row 6, the error in the market clearing condition for the bond is divided by aggregate production. The columns show the mean and the max errors as well as the 0.1st, 10th, 50th, 90th, and 99.9th percentile.

it can be seen that the deep equilibrium net learns better approximations of the policies (first and second rows) and that the errors in the equilibrium conditions decrease (third and fourth rows).

Table 3 reports the mean and maximum relative Euler equation error (in %), error in the KKT conditions ($\times 10^{-2}$), and market clearing error ($\times 10^{-2}$), as well as several percentiles on a simulated path of 10^4 periods, after training the deep equilibrium net for 2×10^5 episodes.[24]. The mean relative error for all equilibrium conditions is below 0.1%. A relative Euler equation error of 0.1% means that the agents, on average, consume 0.1% too much or too little.

[24]We simulate a total of 12,000 periods and discard the first 2000 "burn-in" periods (see, *e.g.,* Krusell and Smith, 1998)

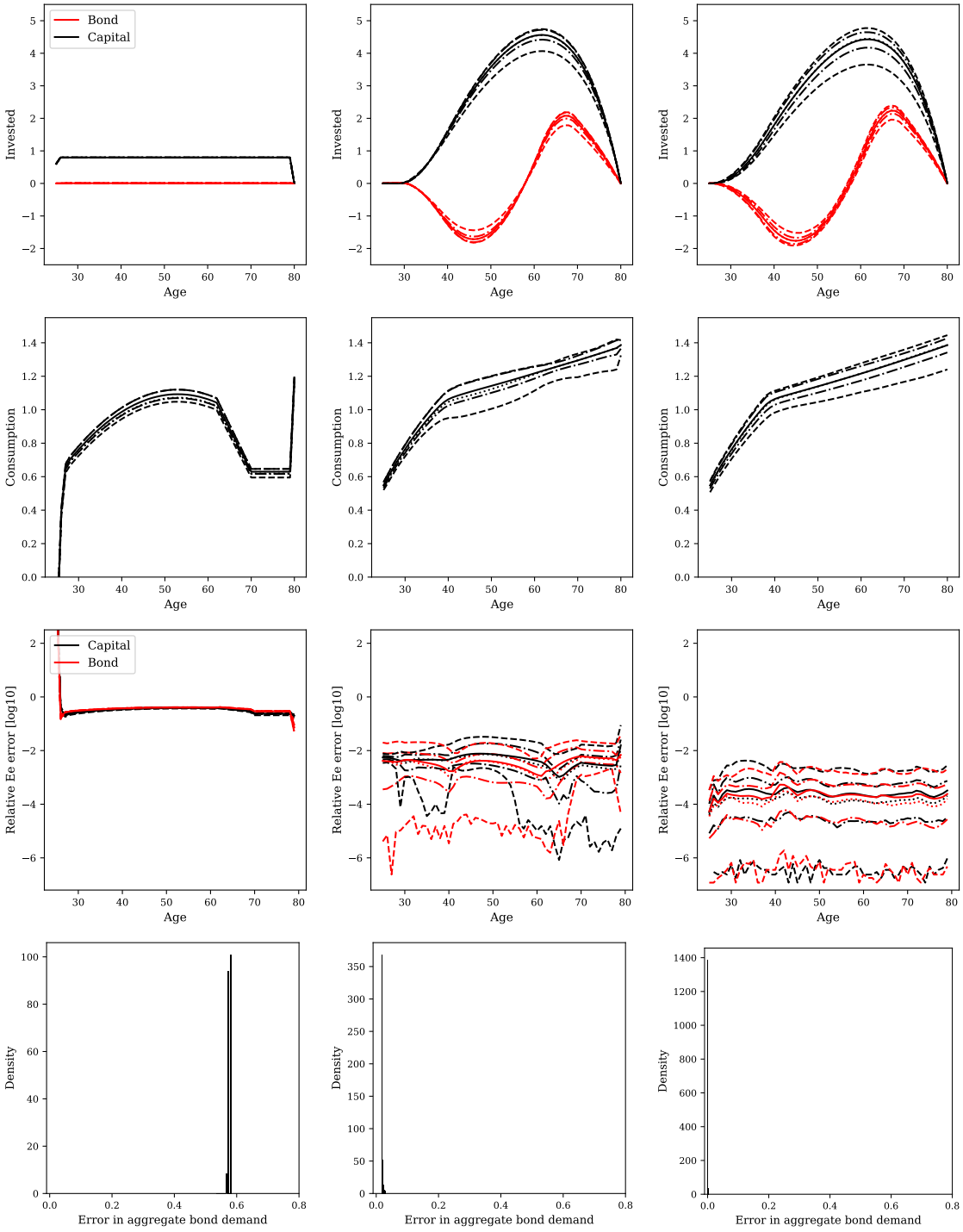


Figure 3: Distribution of investment (first row), consumption (second row), relative Euler equation errors (third row), and errors in the market clearing condition for the bond on a simulated path after learning for 1 (left column), 1000 (middle column), and 200,000 (right column) episodes. The solid line shows the mean over 2000 simulated periods, the dotted line shows the median, the dash-dotted lines show the 10th and 90th percentile, the dashed lines show the 0.1th and 99.9th percentile.

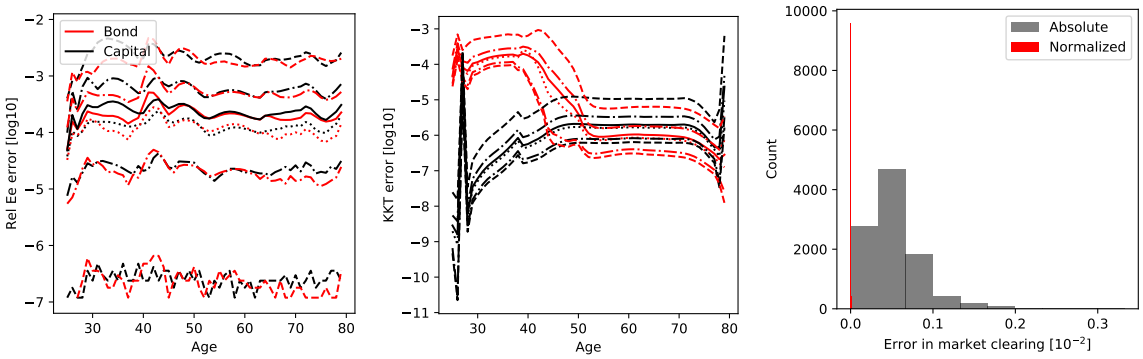


Figure 4: Distribution of the relative Euler equation error (left), error in the KKT conditions (middle), and errors in the market clearing condition (right). The solid line shows the mean over 10,000 simulated periods, the dotted line shows the median, the dash-dotted lines show the 10th and 90th percentile, the dashed lines show the 0.1th and 99.9th percentile.

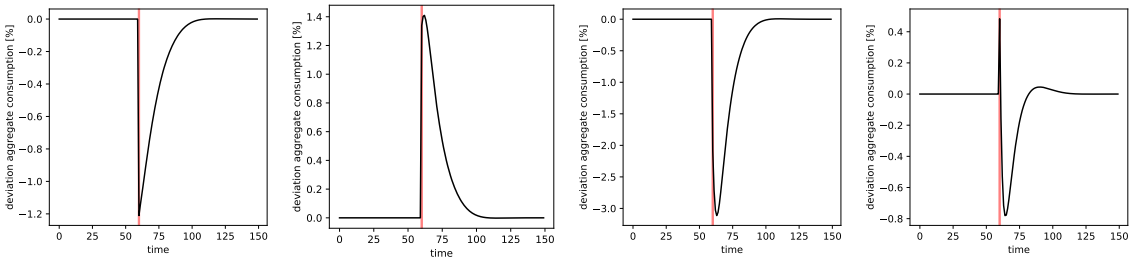
The 99.9th percentile and the maximum of all errors is below 1%. The left and center panel in figure 4 plot the errors across the age-groups and the right panel plots the errors in the market clearing condition. The reached accuracy demonstrates that our method can compute satisfactory approximations to the functional rational expectations equilibrium.

5.3 Consumption response to aggregate shocks

Next, we use the equilibrium functions that we computed in section 5.2 to study the consumption response to aggregate shocks across age-groups. We thereby relate to recent literature, which highlights the importance of the cross-sectional consumption response and the liquidity of assets for understanding the aggregate consumption response to economic shocks (in the context of monetary policy see, *e.g.*, Kaplan et al., 2018; Wong, 2016). First, we analyze the impulse response of aggregate consumption to each of the four shocks. Next, we show how the consumption response to aggregate shocks varies across age-groups.

To study the reaction of consumption to each of the four aggregate shocks, we perform the following exercise. We draw 10^5 starting states from the ergodic distribution.[25] From each starting state, we simulate 150 periods forward twice. In one simulation, the exogenous shock evolves randomly for all 150 periods. In the other, we specify that a particular shock realizes in the period 60. For each time period and each case, with and without a specified shock in the period

[25] We simulate 1000 different stochastic evolutions of the economy with 50,000 periods each, and pick simulated states 500 periods apart as our starting states.



(a) Shock 1: $\eta = 0.978,\ \delta = 0.08$ **(b)** Shock 2: $\eta = 1.022,\ \delta = 0.08$ **(c)** Shock 3: $\eta = 0.978,\ \delta = 0.11$ **(d)** Shock 4: $\eta = 1.022,\ \delta = 0.11$

Figure 5: Mean percent deviation of the aggregate consumption from its unconditional average when each of the exogenous shocks occurs in period 60. The mean impulse response was computed over 100,000 simulations. On impact of the respective shock, aggregate consumption changes as follows: (a) decreases by 1.2%; (b) increases by 1.3%; (c) decreases by 2.1%; and (d) increases by 0.5%, respectively.

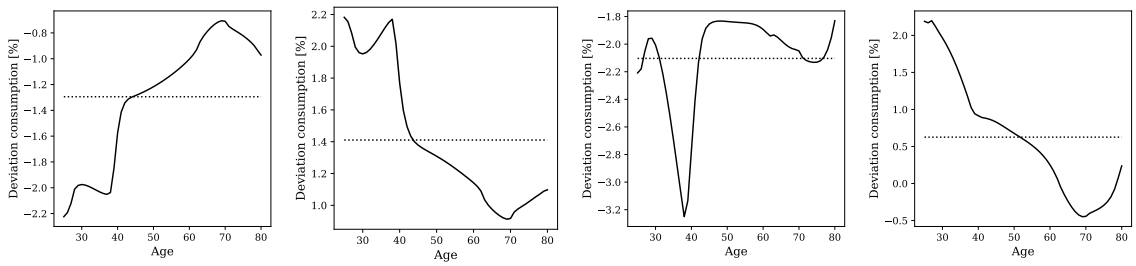
60, we compute averages of the quantities of interest. Then, we compare the time-series of average economic quantities.

First, we study the impulse response of aggregate consumption to each of the four shocks, which is shown in Figure 5. On average, aggregate consumption is about 1.2% lower when the low TFP shock realizes together with normal depreciation and about 1.3% higher when high TFP realizes. In the event of the rare high depreciation shocks, aggregate consumption drops by 2.1% when it realizes together with low TFP and, on impact, increases by 0.5% when it realizes together with high TFP.[26]

Next, we want to understand the response of aggregate consumption by looking at the response across different age-groups. Figure 6 shows the cross-sectional reaction of consumption to each of the four shocks in the period in which the shock realizes. The dotted lines show the mean consumption deviation, averaged across age-groups.

As the figure shows, there is significant heterogeneity across age-groups, where young agents respond the strongest. When the high depreciation shock realizes together with low TFP, these agents earn a low return on their illiquid asset and become borrowing constrained and hence have to reduce their debt, resulting in lower consumption. Agents between 40 and 50 hold a larger level of debt, however they hold enough illiquid wealth so that the collateral constraint rarely binds.

[26]Note that in the latter case, the consumption response is positive on impact, but then negative in the period after.



(a) Shock 1: $\eta = 0.978,\ \delta = 0.08$ **(b)** Shock 2: $\eta = 1.022,\ \delta = 0.08$ **(c)** Shock 3: $\eta = 0.978,\ \delta = 0.11$ **(d)** Shock 4: $\eta = 1.022,\ \delta = 0.11$

Figure 6: Mean percent deviation of consumption by age-group from its unconditional average when each of the exogenous shocks occurs. The dotted lines show the mean across age-groups. The mean is taken over 100,000 simulations.

In order to see the impact of the aggregate shock on cross-sectional consumption in the subsequent periods, figure 10 in appendix A shows the average consumption across age-groups in subsequent periods. The impact of experiencing a high depreciation shock on consumption remains visible for more than 20 years.

6 A model with idiosyncratic shocks

The benchmark model presented in section 3 featured one type of agent and no idiosyncratic risk. Next, we want to enrich the model by adding two sources of idiosyncratic risk over the life-cycle. First, we now consider two possible trajectories of labor endowment over the life-cycle, reflecting two types of agents in the economy. The type of an agent is persistent and realizes at "birth". Second, agents face a health-shock shortly before retiring. Health risks are an important source of uncertainty over the life-cycle, especially in our aging society, and subject to the public and academic debate (see, *e.g.*, De Nardi et al., 2016; Pelgrin and St-Amour, 2016). Hence, we believe that questions related to health insurance are naturally linked to the intergenerational wealth distribution, rendering OLG models a suitable modeling choice. This endorses the need for appropriate solution methods, such as the method we present in this paper. In the stylized model we present in this section, agents experience a reduced utility from consumption in the event of falling sick, causing them to draw heavily on their savings, and hence leaving them a significantly reduced amount of savings when they enter retirement. In this setting, we then study

the welfare gains from introducing health insurance, *i.e.*, the possibility to invest in an additional asset, which pays out contingent on the realization of the health-shock. This section hence serves to illustrate that our method is perfectly applicable in the presence of several idiosyncratic shocks as well as to richer models, which may help address policy-relevant questions.

6.1 The economic model

Except for the two types of agents and the health-shocks, the model remains identical to our benchmark model. Therefore, we introduce the new model with a focus on the differences with respect to the benchmark model, which is described in section 3.

An agent in this economy is indexed by the tuple (y, s, h), and $y \in \{1, 2\}$ denotes the two types of agents. In every cohort, there is a mass one of agents of each type. The agent's age is indexed by $s \in \{1, \dots, N\}$, and $h \in \{1, 2\}$ denotes whether the agent experienced a health-shock ($h = 2$) or not ($h = 1$). The index h evolves according to a type-specific Markov transition probability $\Pi^{(y,s)}$. We assume that the health-shock realizes at fixed age $s = s^h$.[27] The fraction of agents who receive the health-shock at age s^h is given by $\pi^y := \Pi_{1,2}^{(y,s^h-1)}$. Consequently, we have

$$\Pi^{(y,s)} = \mathbb{1}_{s<s^h-1} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \mathbb{1}_{s=s^h-1} \begin{bmatrix} 1-\pi^y & \pi^y \\ 0 & 0 \end{bmatrix} + \mathbb{1}_{s\geq s^h} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{54}$$

We chose s^h to be the last period before retirement. The per-period utility function in this model is given by

$$u\left(\frac{c_t^{(y,s,h)}}{g^{(y,s,h)}}\right), \tag{55}$$

where the factor $g^{(y,s,h)} \geq 1$ models the health-shock. If an agent of type y and age $s = s^h$ experiences the health-shock, *i.e.*, $h = 2$, the agent draws less utility of consumption by a factor $g^{(y,s^h,2)} > 1$. For all other agents, we have $g^{(y,s,h)} = 1$, and the utility from consumption is unchanged. For $s < s^h$, h only takes value 1 for both types. For $s \geq s^h$, we have two agents per type and cohort corresponding to $h = 1$ and $h = 2$.

[27] Since the health-shock only occurs at age s^h, the transition of index h is deterministic for all age-groups other than for age-group $s = s^h - 1$.

The rest of the model follows our benchmark model, which is described in section 3. We denote the agents' holdings in risky, illiquid capital by $k_t^{(y,s,h)}$, the investment is denoted by $a_t^{(y,s,h)}$, where $a_t^{(y,s,h_t)} = k_{t+1}^{(y,s+1,h_{t+1})}$. Similarly, $b_t^{(y,s,h)}$ and $d_t^{(y,s,h)}$ denote bond holdings and bond purchases, respectively, where $d_t^{(y,s,h_t)} = b_{t+1}^{(y,s+1,h_{t+1})}$. The borrowing constraints on capital and the collateral requirement for short-selling the bond remain also as in the benchmark model.

In our benchmark model, there is a single representative agent per age-group, resulting in N heterogeneous agents that interact every period. In contrast, we now have $\tilde{N} = (s^h - 1) + 2 \cdot (N - (s^h - 1))$ agents for each type y. That is, there is one agent per period younger than s^h and two agents per period from age s^h onward, where the members of the latter cohort distinguish themselves through whether they experienced the health-shock. Therefore, for two types $y \in \{1, 2\}$, we model a total of $2 \cdot \tilde{N}$ agents. In our model calibration, $N = 56$ and $s^h = 38$ (corresponding to age 62 when agents are born at 25), resulting in 150 agents and an approximately 300-dimensional endogenous state.[28]

6.2 Functional rational expectations equilibrium

In this section, we define a functional rational expectations equilibrium for the OLG model with idiosyncratic shocks. Note that next to a larger number of agents, the only equilibrium conditions which change are the households' optimality conditions for age-groups $s = s^h - 1$ (*i.e.*, in the period before the health-shock realizes), and $s = s^h$ (*i.e.*, in the period, in which the health-shock realizes). We denote the $\tilde{N}$ policy functions of type y for investment in capital with $a^{(y,s,h)}(\cdot)$, with associated KKT multipliers $\lambda^{(y,s,h)}(\cdot)$, and the policy functions for investment in the bond with $d^{(y,s,h)}(\cdot)$, with associated KKT multipliers $\mu^{(y,s,h)}(\cdot)$. The price function for the bond is denoted by $p(\cdot)$.

Definition 3 (functional rational expectations equilibrium) *A FREE consists of equilibrium functions* $\{a^{(y,s,h)}(\cdot), \lambda^{(y,s,h)}(\cdot), d^{(y,s,h)}(\cdot), \mu^{(y,s,h)}(\cdot), p(\cdot)\}$*, where* $a^{(y,s,h)} : \mathcal{Z} \times \mathbb{R}^{2\tilde{N}} \to \mathbb{R}^{\tilde{N}-2}$ *denotes the capital investment functions,* $\lambda^{(y,s,h)} : \mathcal{Z} \times \mathbb{R}^{2\tilde{N}} \to \mathbb{R}^{\tilde{N}-2}$ *denotes the KKT-multiplier functions for the short-selling constraint on capital,* $d^{(y,s,h)} : \mathcal{Z} \times \mathbb{R}^{2\tilde{N}} \to \mathbb{R}^{\tilde{N}-2}$ *denotes the bond investment*

[28] The endogenous state consists of the asset holdings across age-groups. For 150 agents and two assets, this results in a 300-dimensional endogenous state-space. Since agents are born without assets, and the beginning-of-period asset holdings of agents aged s^h are the same within a type, irrespective of the health-shock, a sufficient endogenous state could be 292-dimensional.

functions, and $\mu^{(y,s,h)} : \mathcal{Z} \times \mathbb{R}^{2\tilde{N}} \to \mathbb{R}^{\tilde{N}-2}$ *denotes the KKT-multiplier functions for the collateral constraint for type* $y \in \{1,\ 2\}$*, such that for all states* $\mathbf{x} := [z,\ \mathbf{k}^{1T},\ \mathbf{k}^{2T},\ \mathbf{b}^{1T},\ \mathbf{b}^{2T}]^T \in \mathcal{Z} \times \mathbb{R}^{2\tilde{N}}$*, where* $z \in \mathcal{Z}$ *denotes the exogenous shock, and the capital holding* $\mathbf{k}^y$ *together with the bond holding* $\mathbf{b}^y$ *denote the endogenous state (i.e., the distribution of capital holdings and bond holdings) with* $k^{(y,1,1)} = 0$ *and* $b^{(y,1,1)} = 0$*, we have for all* $s = 1, \ldots, N-1$*,* $y = 1, 2$*,* $h = 1$ *for* $s < s^h$ *and* $h = 1, 2$ *for* $s \geq s^h$*:*

$$\begin{aligned}
&\frac{1+\zeta\Delta^{(y,s,h)}(\mathbf{x})}{g^{(y,s,h)}}u'\left(\frac{c^{(y,s,h)}(\mathbf{x})}{g^{(y,s,h)}}\right) \\
&\quad = \beta E_{z,h}\left[\frac{r(\mathbf{x}_+)\left(1+\zeta\Delta^{(y,s+1,h_+)}(\mathbf{x}_+)\right)}{g^{(y,s+1,h_+)}}u'\left(\frac{c^{(y,s+1,h_+)}(\mathbf{x}_+)}{g^{(y,s+1,h_+)}}\right)\right] \\
&\qquad + \lambda^{(y,s,h)}(\mathbf{x}) + \mu^{(y,s,h)}(\mathbf{x}) && (56)\\
&0 = a^{(y,s,h)}(\mathbf{x}) \cdot \lambda^{(y,s,h)}(\mathbf{x}) && (57)\\
&0 \leq a^{(y,s,h)}(\mathbf{x}) && (58)\\
&0 \leq \lambda^{(y,s,h)}(\mathbf{x}) && (59)
\end{aligned}$$

as well as

$$\begin{aligned}
&\frac{p(\mathbf{x})}{g^{(y,s,h)}}u'\left(\frac{c^{(y,s,h)}(\mathbf{x})}{g^{(y,s,h)}}\right) \\
&\quad = \quad \beta E_{z,h}\left[\frac{\min\{\kappa r(\mathbf{x}_+), 1\}}{g^{(y,s+1,h_+)}}u'\left(\frac{c^{(y,s+1,h_+)}(\mathbf{x}_+)}{g^{(y,s+1,h_+)}}\right)\right] \\
&\qquad + \kappa\mu^{(y,s,h)}(\mathbf{x}) && (60)\\
&0 = \mu^{(y,s,h)}(\mathbf{x})\left(a^{(y,s,h)}(\mathbf{x}) + \kappa d^{(y,s,h)}(\mathbf{x})\right) && (61)\\
&0 \leq a^{(y,s,h)}(\mathbf{x}) + \kappa d^{(y,s,h)}(\mathbf{x}) && (62)\\
&0 \leq \mu^{(y,s,h)}(\mathbf{x}) && (63)
\end{aligned}$$

and

$$0 = \sum_{y=1,2}\left(\sum_{s=1}^{s^h-1} d^{(y,s,1)}(\mathbf{x}) + \sum_{s=s^h}^{N-1}(1-\pi^y)d^{(y,s,1)}(\mathbf{x}) + \pi^y d^{(y,s,2)}(\mathbf{x})\right), \qquad (64)$$

where $r(\mathbf{x}) = f_K(K, L, x_1)$ *and* $w(\mathbf{x}) = f_L(K, L, x_1)$*, with*

$$K = \sum_{y=1,2} \left(\sum_{s=1}^{s^h-1} k^{(y,s,1)}(\mathbf{x}) + \sum_{s=s^h}^{N} (1-\pi^y) k^{(y,s,1)}(\mathbf{x}) + \pi^y k^{(y,s,2)}(\mathbf{x}) \right), \quad (65)$$

$$L = \sum_{y=1,2} \left(\sum_{s=1}^{s^h-1} l^{(y,s,1)}(\mathbf{x}) + \sum_{s=s^h}^{N} (1-\pi^y) l^{(y,s,1)}(\mathbf{x}) + \pi^y l^{(y,s,2)}(\mathbf{x}) \right). \quad (66)$$

The budget constraint remains unchanged.

As before in our benchmark model, computing an approximation to the FREE in this model means finding an approximation for the $4 \cdot (\tilde{N} - 2) + 1$ equilibrium functions such that the above equations are approximately fulfilled for all exogenous shocks and all values the $2\tilde{N}$-dimensional endogenous state takes. Besides the larger number of agents and, as a result, the larger number of functions to be approximated and the larger number of equations to be fulfilled, the algorithm remains the same.

6.2.1 Insurance

To study the welfare implication of health insurance in this model, we add additional assets to the model, which pay out contingent on the realization of the health-shock. Agents of type y at age $s^h - 1$ can buy an asset, which promises a payout of 1 in case the health-shock hits the agent in the following period. The collateral requirements are the same as for the bond. From the perspective of an agent that sells insurance, selling one unit of insurance is indistinguishable from selling π^y units of the bond. Hence, the price $p^y(\mathbf{x})$ of one unit of insurance for type y is pinned down by the bond price and is given by[29]

$$p^y(\mathbf{x}) = \pi^y p(\mathbf{x}). \quad (67)$$

Let $i^y(\mathbf{x})$ denote the units of insurance purchased by age-group $s^h - 1$, then

[29] We can also see the purchase of insurance the following way: the representative agent of type y at age $s^h - 1$ buys π^y units of the bond. The fraction $(1 - \pi^y)$ of households, which don't fall sick the next period, promises the payout of that bond to the fraction π^y of households, which get sick and then receive a payout of $\pi^y + \pi^y \frac{1-\pi^y}{\pi^y} = 1$.

the new bond market clearing condition reads

$$0 = \sum_{y=1,2} \left(\pi^y i^y(\mathbf{x}) + \sum_{s=1}^{s^h-1} d^{(y,s,1)}(\mathbf{x}) + \sum_{s=s^h}^{N-1} (1-\pi^y) d^{(y,s,1)}(\mathbf{x}) + \pi^y d^{(y,s,2)}(\mathbf{x}) \right). \tag{68}$$

The amount of insurance, $i^y(\mathbf{x})$, purchased by an agent of type y and age $s^y - 1$ is pinned down by an additional Euler equation

$$p^y(\mathbf{x}) u'\left(c^{(y,s^h-1,1)}(\mathbf{x})\right) = \beta \mathrm{E}_z \left[\pi^y \frac{\min\{\kappa r(\mathbf{x}_+), 1\}}{g^{(y,s^h,2)}} u'\left(\frac{c^{(y,s^h,2)}(\mathbf{x}_+)}{g^{(y,s^h,2)}} \right) \right]. \tag{69}$$

Here, we do not include any multipliers because we assume that the short-selling constraint on the health insurance never binds. Therefore, we add the relative error in equation (69) directly to the cost function. Note that the cost function for this model has the same form as the cost function of the benchmark model shown in equation (49), *i.e.*, it is given by the mean-squared errors in the relative Euler equations for bond, capital, and insurance as well as the market clearing condition for the bond.

6.3 Model parameter setup

In this section, we introduce the parameters, which are new in the OLG model with idiosyncratic shocks. We model $N = 56$ age-groups, the health-shock occurs at age 62. There are now two, type-specific, profiles of labor endowment over the life-cycle (see, fig. 9 in appendix A). The labor profile of type 1 starts lower, rises steeper, and peaks higher. The labor endowment of type $y = 1$ rises by 180 % between ages 25 and its peak value at age 53, whereas the labor endowment of type $y = 2$ rises by 87 %. The average labor endowment by age-group rises by approximately 127 %, as in the benchmark model. We chose the probability of the health-shock to be $\pi^1 = 0.1$ for type 1 and $\pi^2 = 0.3$ for type 2. The health-shock is also more severe for type 2 than for type 1, with $g^{(1,s^h,2)} = 2$ and $g^{(2,s^h,2)} = 3$. The remaining parameters stay the same as in the benchmark model (a summary is given in table 4).

Discount factor β	Relative risk aversion γ	Capital share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1}=1.022\|\eta_t=1.022)$ $P(\eta_{t+1}=0.978\|\eta_t=0.978)$	Persistence depreciation $P(\delta_{t+1}=0.08\|\delta_t=0.08)$ $P(\delta_{t+1}=0.11\|\delta_t=0.11)$	Borrowing constraint $\underline{a}$	Adjustment cost capital ζ	Collateral requirement bond κ	Timing of health shock s^h	Probability of health shock π^1 π^2	Magnitude of health shock $g^{(1,s^h,2)}$ $g^{(1,s^h,2)}$
0.95	2	0.3	$\{0.978, 1.022\}$	$\{0.08, 0.11\}$	0.905 0.905	0.972 0.700	0.0	0.5	1.1236	38	0.1 0.3	2 3

Table 4: Parameterization of the model with two types and health-shocks (*cf.* sec. 6.1).

6.4 Performance of the deep equilibrium net

For both cases, the model with and without health-insurance, we train the neural network for 100,000 episodes. The chosen hyper-parameters are shown in table 7 in appendix B. Tables 8 and 9 in appendix B, show statistics for the errors in the equilibrium conditions on 10,000 simulated periods after the training was completed. As in the benchmark model, the mean and maximum relative errors in the Euler and KKT conditions remain below 0.1% and 1% respectively.

6.5 Welfare benefits from health insurance

We now make use of our approximated model solution to study the welfare benefits, which arise from the possibility to buy health-insurance, *i.e.*, an asset with a payoff conditional on receiving the health-shock in the context of the model presented in section 6.1. To do so, we compare the mean expected utility in the setting with and without insurance. We compare the expected lifetime utility before the agents enter the economy, *i.e.*, before their type has realized, to the expected lifetime utility conditional on the type and conditional on receiving or not receiving the health-shock. We convert the difference in lifetime utility to the fraction of consumption an agent would additionally need to receive in every state of the world in order to attain the same lifetime utility in both cases. This factor, denoted by q, can be computed noting that for CRRA utility with coefficient of relative risk aversion γ, the expected lifetime utility scales by a factor $(1+q)^{1-\gamma}$ if consumption in every state is scaled by a factor $(1+q)$: $V((1+q)c) = (1+q)^{1-\gamma}V(c)$.

Table 5 shows the lifetime utility in the economy without insurance. As shown in table 5, type 2 agents who do not receive the health-shock attain the highest mean lifetime utility. The difference to the unconditional average corresponds to approximately 2.1% of consumption. Furthermore, the table shows that the type of shock at birth has a larger influence on the mean attained lifetime utility than the health-shock. The difference in mean received lifetime

	Unconditional	Cond. on type					
		T1	T2	Cond. on health shock			
				T1, sick	T1, healthy	T2, sick	T2, healthy
Mean lifetime utility	-18.206	-18.533	-17.878	-18.609	-18.524	-17.992	-17.830
Consumption factor [%] (compared to unconditional)	0.0	-1.8	+1.8	-2.2	-1.7	+1.2	+ 2.1

Table 5: Mean lifetime utility in the model where no insurance is available, computed from 100,000 simulations of 150 time periods. The first row shows the mean lifetime utility; the second row shows the percentage of extra consumption the mean agent would need to receive in order to attain the same mean lifetime utility as lucky/unlucky agents, who received specific shocks.

	Unconditional	Cond. on type					
		T1	T2	Cond. on health shock			
				T1, sick	T1, healthy	T2, sick	T2, healthy
Mean lifetime utility	-18.204	-18.531	-17.877	-18.569	-18.527	-17.933	-17.852
Consumption factor [%] (compared to unconditional)	0.0	-1.8	+1.8	-2.0	-1.7	+1.5	+ 2.0

Table 6: Mean lifetime utility in the model where insurance is available, computed from 100,000 simulations of 150 time periods. The first row shows the mean lifetime utility. The second row shows the percentage of extra consumption the mean agent would need to receive in order to attain the same mean lifetime utility as lucky/unlucky agents, who received specific shocks.

utility between agents born as type 1 and agents born as type 2 corresponds to 3.6% of consumption. The difference between receiving and not receiving the shock is approximately 0.5% and 0.9% for type 1 and 2, respectively.

Table 6 shows the same numbers for the economy in which health insurance is available. The effect on mean attained lifetime utility is not significant when looking at the unconditional average, or the average conditional on the type. This may be due to the fact that most of the lifetime utility is accumulated in earlier years, before the health-shock hits. Since the health-shock occurs in the 38th modeled period (at age 62), utility attained in the remaining periods is discounted with $\beta^{37} \approx 0.15$ and more when computing attained lifetime utility.

The difference in attained lifetime utility between agents who receive the health-shock and those who do not is halved through the availability of insurance. Focusing on the utility accumulated from age 62 (the age at which the health-shock occurs) and onwards, the difference in utility attained by agents who fell sick and who stayed healthy corresponds to -6.6% of consumption for type 1 agents and to -11.2% of consumption for type 2 agents in the model without health insurance. In the model with health insurance, the differences reduce to -3.4% and -5.8%. In our stylized model, the ability to buy health insurance hence has the most significant impact on older agents.

7 Conclusion

In this paper, we introduce deep equilibrium nets—neural networks that approximate all equilibrium functions and are trained on simulated data to satisfy all equilibrium conditions. Since the neural network approximates the equilibrium functions directly, neither sets of non-linear equations nor optimization problems need to be solved in order to simulate the economy. Consequently, training data can be generated at virtually zero cost, which allows us to swiftly train the neural network on more than a billion simulated states of the economy. Deep equilibrium nets thus provide a grid-free, global, solution method, which can jointly address the computational challenges arising from a high-dimensional state-space, significant uncertainty, financial frictions, and irregular geometries of the state-space. The training process is fully parallelizable, which allows efficient use of contemporary high-performance computing hardware.

We illustrate the performance of our method in the context of two different OLG models with a 108 and 292-dimensional endogenous state-space, occasionally binding constraints, and aggregate uncertainty. Both models feature two types of assets: risky, illiquid capital, and a risk-free, one-period bond subject to collateral requirements. In the benchmark model, we study the aggregate and cross-sectional consumption response to aggregate shocks and find significant heterogeneity across age-groups. The second model additionally features two types of agents and an idiosyncratic health-shock. We use the latter setting to study how the availability of health insurance affects conditional mean lifetime utility. In this stylized model, health insurance reduces the difference in mean lifetime utility between agents who remained healthy and agents who fell sick by roughly a factor of two.

Both examples show that our proposed method can accurately approximate recursive equilibria in economically relevant settings with a high-dimensional state-space, uncertainty, and strong non-linearities.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Lev-

enberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aruoba, S. B., Fernandez-Villaverde, J., and Rubio-Ramirez, J. F. (2006). Comparing solution methods for dynamic equilibrium economies. *Journal of Economic dynamics and Control*, 30(12):2477–2508.

Becker, S., Cheridito, P., and Jentzen, A. (2018). Deep optimal stopping. *arXiv preprint arXiv:1804.05394*.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

Bilbiie, F. O. (2008). Limited asset markets participation, monetary policy and (inverted) aggregate demand logic. *Journal of Economic Theory*, 140(1):162 – 196.

Boppart, T., Krusell, P., and Mitman, K. (2018). Exploiting MIT shocks in heterogeneous-agent economies: the impulse response as a numerical derivative. *Journal of Economic Dynamics and Control*, 89:68 – 92. Fed St. Louis-JEDC-SCG-SNB-UniBern Conference, titled: "Fiscal and Monetary Policies".

Brumm, J., Kubler, F., and Scheidegger, S. (2017). Computing equilibria in dynamic stochastic macro-models with heterogeneous agents. In *Advances in Economics and Econometrics: Volume 2: Eleventh World Congress*, volume 59, page 185. Cambridge University Press.

Brumm, J. and Scheidegger, S. (2017). Using adaptive sparse grids to solve high-dimensional dynamic models. *Econometrica*, 85(5):1575–1612.

Chen, X. and White, H. (1999). Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory*, 45(2):682–691.

De Nardi, M., French, E., and Jones, J. B. (2016). Medicaid insurance in old age. *American Economic Review*, 106(11):3480–3520.

Debortoli, D. and Galí, J. (2017). Monetary policy with heterogeneous agents: Insights from tank models. *Manuscript, September*.

Den Haan, W. J. (2010). Assessing the accuracy of the aggregate law of motion in models with heterogeneous agents. *Journal of Economic Dynamics and Control*, 34(1):79–99.

Den Haan, W. J. and Marcet, A. (1990). Solving the stochastic growth model by parameterizing expectations. *Journal of Business & Economic Statistics*, 8(1):31–34.

Dou, W. W., Fang, X., Lo, A. W., and Uhlig, H. (2017). Comparing solution methodologies for macro-finance models with nonlinear dynamics. Working paper.

Duarte, V. (2018a). Machine learning for continuous-time economics. Working paper.

Duarte, V. (2018b). Sectoral reallocation and endogenous risk-aversion: Solving macro-finance models with machine learning.

Duffy, J. and McNelis, P. D. (2001). Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm. *Journal of Economic Dynamics and Control*, 25(9):1273–1303.

Fernández-Villaverde, J., Gordon, G., Guerrón-Quintana, P., and Rubio-Ramirez, J. F. (2015). Nonlinear adventures at the zero lower bound. *Journal of Economic Dynamics and Control*, 57:182–204.

Fernandez-Villaverde, J. and Guerron-Quintana, P. (2020). Uncertainty Shocks and Business Cycle Research. *Review of Economic Dynamics*, 37:118–166.

Fernández-Villaverde, J., Hurtado, S., and Nuno, G. (2019). Financial frictions and the wealth distribution. Working paper.

Fernández-Villaverde, J., Rubio-Ramírez, J., and Schorfheide, F. (2016). Chapter 9 - solution and estimation methods for dsge models. volume 2 of *Handbook of Macroeconomics*, pages 527 – 724. Elsevier.

Gaspar, J. and Judd, K. L. (1997). Solving large-scale rational-expectations models. *Macroeconomic Dynamics*, 1(1):45–75.

Gervais, M., Jaimovich, N., Siu, H. E., and Yedid-Levi, Y. (2016). What should i be when i grow up? occupations and unemployment over the life cycle. *Journal of Monetary Economics*, 83:54 – 70.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gourio, F. (2012). Disaster risk and business cycles. *American Economic Review*, 102(6):2734–66.

Grohs, P., Hornung, F., Jentzen, A., and Von Wurstemberger, P. (2018). A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362*.

Guvenen, F., Karahan, F., Ozkan, S., and Song, J. (2015). What do data on millions of us workers reveal about life-cycle earnings risk? Technical report, National Bureau of Economic Research.

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

Huffman, G. W. (1987). A dynamic equilibrium model of asset prices and transaction volume. *Journal of Political Economy*, 95(1):138–159.

Hutchinson, J. M., Lo, A. W., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.

Jentzen, A., Salimova, D., and Welti, T. (2018). A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*.

Judd, K. L. (1992). Projection methods for solving aggregate growth models. *Journal of Economic Theory*, 58(2):410–452.

Judd, K. L. (1998). *Numerical methods in economics*. The MIT press.

Judd, K. L., Maliar, L., and Maliar, S. (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. *Quantitative Economics*, 2(2):173–210.

Judd, K. L., Maliar, L., Maliar, S., and Valero, R. (2014). Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123.

Juillard, M. (1996). Dynare : a program for the resolution and simulation of dynamic models with forward variables through the use of a relaxation algorithm. CEPREMAP Working Papers (Couverture Orange) 9602, CEPREMAP.

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

Keane, M. P. and Wolpin, K. I. (1994). The solution and estimation of discrete

choice dynamic programming models by simulation and interpolation: Monte carlo evidence. *The Review of Economics and Statistics*, 76(4):648–672.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kollmann, R., Maliar, S., Malin, B. A., and Pichler, P. (2011). Comparison of solutions to the multi-country real business cycle model. *Journal of Economic Dynamics and Control*, 35(2):186–202.

Kotlikoff, L., Kubler, F., Polbin, A., Sachs, J., and Scheidegger, S. (2020a). Making carbon taxation a generational win win. *International Economic Review*.

Kotlikoff, L. J., Kubler, F., Polbin, A., and Scheidegger, S. (2020b). Pareto-improving carbon-risk taxation. Working Paper 26919, National Bureau of Economic Research.

Krueger, D. and Kubler, F. (2004). Computing equilibrium in olg models with stochastic production. *Journal of Economic Dynamics and Control*, 28(7):1411 – 1436.

Krueger, D. and Kubler, F. (2006). Pareto-improving social security reform when financial markets are incomplete!? *American Economic Review*, 96(3):737–755.

Krueger, D., Mitman, K., and Perri, F. (2016). Chapter 11 - macroeconomics and household heterogeneity. volume 2 of *Handbook of Macroeconomics*, pages 843 – 921. Elsevier.

Krusell, P. and Smith, Jr, A. A. (1998). Income and Wealth Heterogeneity in the Macroeconomy. *Journal of Political Economy*, 106(5):867–896.

Kubler, F. and Scheidegger, S. (2018). Self-justified equilibria: Existence and computation.

Kubler, F. and Schmedders, K. (2003). Stationary equilibria in asset-pricing models with incomplete markets and collateral. *Econometrica*, 71(6):1767–1793.

Lepetyuk, V., Maliar, L., and Maliar, S. (2019). When the u.s. catches a cold, canada sneezes: A lower-bound tale told by deep learning. CEPR Discussion Paper DP14025.

Ljungqvist, L. and Sargent, T. (2000). *Recursive macroeconomic theory*. Mit Press.

Magill, M. J. (1977). A local analysis of n-sector capital accumulation under uncertainty. *Journal of Economic Theory*, 15(1):211–219.

Maliar, L. and Maliar, S. (2014). Numerical methods for large-scale dynamic

economic models. In *Handbook of computational economics*, volume 3, pages 325–477. Elsevier.

Maliar, L. and Maliar, S. (2015). Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model. *Quantitative Economics*, 6(1):1–47.

Maliar, L., Maliar, S., and Winant, P. (2019). Will artificial intelligence replace computational economists any time soon? CEPR Discussion Paper DP14024.

Montanelli, H. and Du, Q. (2019). New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

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

Pelgrin, F. and St-Amour, P. (2016). Life cycle responses to health insurance status. *Journal of Health Economics*, 49:76 – 96.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

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

Renner, P. and Scheidegger, S. (2018). Machine learning for dynamic incentive problems. Working paper.

Rouwenhorst, K. (1995). Asset pricing implications of equilibrium business cycle models. Chapter 10, Frontiers of Business Cycle Research, T. Cooley.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

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

Scheidegger, S. and Treccani, A. (2018). Pricing american options under high-dimensional models with recursive adaptive sparse expectations. *Journal of Financial Econometrics*.

Sergeev, A. and Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm

for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.

Spear, S. E. (1988). Existence and local uniqueness of functional rational expectations equilibria in dynamic economic models. *Journal of Economic Theory*, 44(1):124–155.

Uhlig, H. (1998). A toolkit for analysing nonlinear dynamic stochastic models easily.

Usui, T. (2019). Adaptation to rare natural disasters and global sensitivity analysis in a dynamic stochastic economy. *Available at SSRN 3462011*.

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

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

Wong, A. (2016). Transmission of monetary policy to consumption and population aging. Technical report, mimeo, April 21, Princeton University, Princeton.

Appendix

A Supplementary figures

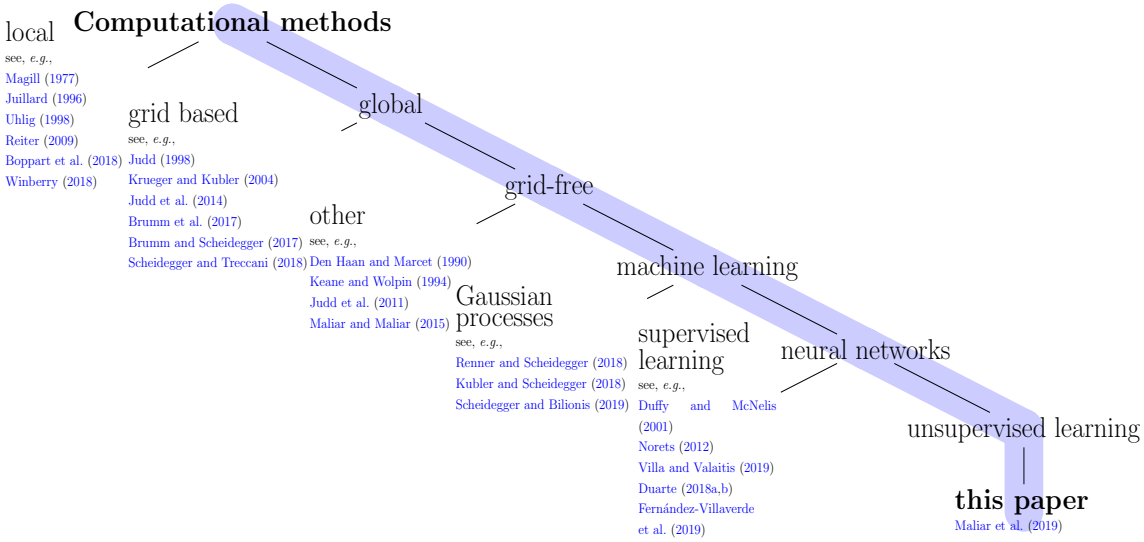


Figure 7: Illustration of the placement of this paper within the literature.

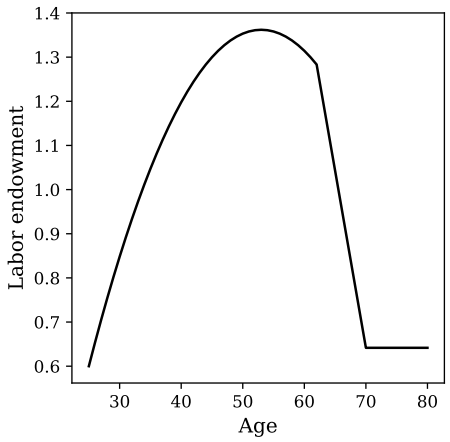


Figure 8: Labor endowment over the life-cycle in the benchmark model.

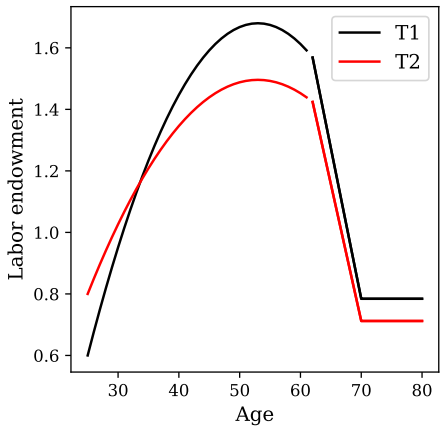
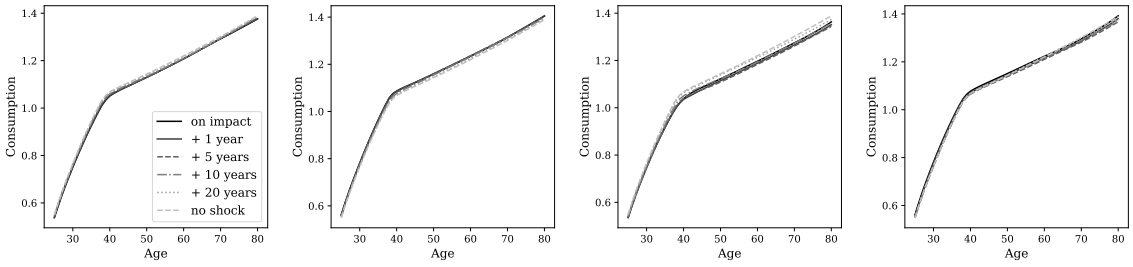


Figure 9: Labor endowment over the life-cycle for type $l = 1$ (in black) and type $l = 2$ (in red) in the models with health shocks. The health-shock occurs at age 62, the year after the line-break.



(a) Shock 1: $\eta = 0.978,\ \delta = 0.08$ **(b)** Shock 2: $\eta = 1.022,\ \delta = 0.08$ **(c)** Shock 3: $\eta = 0.978,\ \delta = 0.11$ **(d)** Shock 4: $\eta = 1.022,\ \delta = 0.11$

Figure 10: Mean consumption across age-groups after realization of an aggregate shock. The mean is taken over 100,000 simulations.

B Supplementary tables

Episodes	Learning rate α^{learn}	Periods per episode T	Epochs per episode N^{epochs}	Mini-batch size m	Nodes hidden layers	Activations hidden layers
1 – 20,000	1×10^{-5}	10,000	1	64	1,000	relu
					1,000	relu
20,001 – 100,000	1×10^{-6}	10,000	1	1,000	1,000	relu
					1,000	relu

Episodes	Learning rate α^{learn}	Periods per episode T	Epochs per episode N^{epochs}	Mini-batch size m	Nodes hidden layers	Activations hidden layers
1 – 18,500	1×10^{-5}	10,000	1	64	1,000	relu
					1,000	relu
18.501 – 100,000	1×10^{-6}	10,000	1	1,000	1,000	relu
					1,000	relu

Table 7: Hyper-parameters chosen to train the deep equilibrium net for the model with two types and health-shocks. The top table shows the case without insurance, the bottom table shows the case with insurance.

	mean	max	0.1	10	50	90	99.9
T1, Rel Ee capital [%]	0.025	0.533	0.000	0.002	0.013	0.060	0.310
T1, Rel Ee bond [%]	0.021	0.554	0.000	0.002	0.011	0.050	0.257
T1, KKT capital [$\times 10^{-2}$]	0.001	0.132	0.000	0.000	0.000	0.001	0.039
T1, KKT bond [$\times 10^{-2}$]	0.003	0.150	0.000	0.000	0.000	0.011	0.052
T2, Rel Ee capital [%]	0.028	0.844	0.000	0.002	0.014	0.068	0.345
T2, Rel Ee bond [%]	0.024	0.881	0.000	0.002	0.012	0.057	0.321
T2, KKT capital [$\times 10^{-2}$]	0.000	0.140	0.000	0.000	0.000	0.001	0.013
T2, KKT bond [$\times 10^{-2}$]	0.004	0.196	0.000	0.000	0.000	0.013	0.072
Market clearing [$\times 10^{-2}$]	0.193	1.679	0.001	0.056	0.169	0.362	1.003
Market clearing (normalized) [$\times 10^{-2}$]	0.000	0.004	0.000	0.000	0.000	0.001	0.002

Table 8: Errors in the equilibrium conditions for the model without health-insurance. The first four rows show the equilibrium conditions for type 1 agents, rows 5 to 8 show the conditions for type 2 agents. The first two rows show the relative Euler equation errors (Rel Ee in %) on 10,000 simulated periods. The next two rows show the errors in the KKT conditions ($\times 10^{-2}$). Rows 9 and 10 show the errors in the market clearing conditions ($\times 10^{-2}$). In row 10, the error in the market clearing condition for the bond is divided by aggregate production. The columns show the mean and the max errors as well as the 0.1st, 10th, 50th, 90th, and 99.9th percentile.

	mean	max	0.1	10	50	90	99.9
T1, Rel Ee capital [%]	0.027	0.522	0.000	0.002	0.014	0.066	0.319
T1, Rel Ee bond [%]	0.022	0.467	0.000	0.002	0.012	0.053	0.253
T1, Rel Ee insurance [%]	0.016	0.196	0.000	0.003	0.012	0.032	0.113
T1, KKT capital [$\times 10^{-2}$]	0.001	0.049	0.000	0.000	0.000	0.001	0.038
T1, KKT bond [$\times 10^{-2}$]	0.003	0.135	0.000	0.000	0.000	0.011	0.058
T2, Rel Ee capital [%]	0.029	0.820	0.000	0.003	0.016	0.070	0.376
T2, Rel Ee bond [%]	0.025	0.847	0.000	0.002	0.014	0.061	0.330
T2, Rel Ee insurance [%]	0.024	0.030	0.000	0.004	0.017	0.055	0.162
T2, KKT capital [$\times 10^{-2}$]	0.000	0.178	0.000	0.000	0.000	0.001	0.003
T2, KKT bond [$\times 10^{-2}$]	0.004	0.000	0.000	0.000	0.000	0.012	0.079
Market clearing [$\times 10^{-2}$]	0.723	2.303	0.022	0.561	0.731	0.880	1.666
Market clearing (normalized) [$\times 10^{-2}$]	0.002	0.005	0.000	0.001	0.002	0.002	0.004

Table 9: Errors in the equilibrium conditions for the model with health-insurance. The first five rows show the equilibrium conditions for type 1 agents, rows 6 to 9 show the conditions for type 2 agents. The first three rows show the relative Euler equation errors (Rel Ee in %) on 10,000 simulated periods. The next two rows show the errors in the KKT conditions ($\times 10^{-2}$). Rows 11 and 12 show the errors in the market clearing conditions ($\times 10^{-2}$). In row 12, the error in the market clearing condition for the bond is divided by aggregate production. The columns show the mean and the max errors as well as the 0.1st, 10th, 50th, 90th, and 99.9th percentile.

C Background on neural networks

C.1 Mini-batches, stochastic gradient descent, and the learning rate

Gradient descent is a first-order optimization method in which a loss function is minimized by updating an algorithm's parameters in the decreasing direction of the loss function's gradient. There are many variations and extensions to the gradient descent optimization algorithm that help accelerate and stabilize training. We briefly outline those used in this paper: namely, the learning rate, mini-batch gradient descent, and the Adam optimizer.

The learning rate, which corresponds to the step-size taken relative to the gradient (see eq. (50)), is one hyper-parameter that can be tuned to facilitate training. We found it helpful to use a larger learning rate at the beginning of training and decrease it towards the end. That is, a large learning rate allows for the quick traversal of the loss function while the parameters are "far" from the optimum. A small learning rate enables a tighter convergence around the optimum. By setting the learning rate too high, we can overshoot and oscillate around the optimum or land in a region where the parameters generate economically nonsensical values (see appendix D.3). Contrarily, by

setting the learning rate too low, training can be unnecessarily slowed or halted entirely if non-linear features of the loss function cannot be overcome.

Mini-batch stochastic gradient descent is another method that can be used. Rather than computing the gradients on the full dataset, which can be computationally expensive, the gradients are estimated on a subset of randomly sampled data (without replacement), *i.e.*, on a *mini-batch*. With mini-batches, the estimates of the gradients are computationally cheaper than with gradient descent. Furthermore, we can perform multiple updates of the network parameters in a single epoch. Finally, the variance of the estimates can be controlled by fine-tuning the mini-batch size. That is, the smaller the mini-batches, the higher the variance of the estimates.

Additionally, various gradient descent optimizers stabilize the training of neural networks by continually adapting the learning rate and gradient.[30] To this end, the Adam optimizer (Kingma and Ba, 2014) uses decayed momentum and a decayed adaptive learning rate. That is, *momentum* updates the parameters using a weighted moving average of the previous and current gradients, thereby favoring steps that occur in a consecutively consistent direction. In Adam, the learning rates are adapted through the normalization by a weighted average of the gradient's second moment.

D Practical issues

D.1 Practical issues of deep equilibrium nets

In this section, we briefly describe some of the challenges our solution method faces in practical applications. A significant drawback of deep neural networks is, in general, that both the theoretical converge rates as well as why and when deep neural networks can ameliorate or break the curse of dimensionality are poorly understood. Even though substantial progress has recently been made (see, *e.g.*, Montanelli and Du, 2019), methods such as ours still heavily depend on numerical experimentation. Such shortcomings are consequently reflected in our solution framework.

A key issue common to the field of deep learning is that the choice of hyperparameters is crucial for the success of a learning algorithm. Nevertheless, the choice is often *ad hoc* or based on prior research experience. The develop-

[30] See, *e.g.*, Ruder (2016) for an extensive overview of neural network optimization methods.

ment of frameworks to automate the process of hyper-parameter optimization is underway and may help mitigate this issue in future research.

Our proposed algorithm is not guaranteed to converge to a satisfactory solution. While we often obtain very good results, we also encountered instances for which the algorithm seems to converge despite significant errors in the equilibrium conditions. In this regard, the iteration between stochastic simulation and variants of mini-batch gradient descent on simulated data may pose a particular challenge: if the simulated states happen to be too far from states the neural network was previously trained on, the approximation on the simulated states may be poor and thus lead to very large Euler equation errors or punishment-factors due to, for example, policies implying negative consumption. When evaluated on such states, the gradients of the loss function with respect to the neural network's parameters may be so large that all the previous learning progress is overwritten. This may lead to vastly different policies, which in turn, may amplify the problem as consecutively simulated states may be far from the states the neural network was trained on previously. In some instances, this problem can be mitigated by choosing a lower learning rate or larger batch-sizes. Alternatively, in some cases, it may be possible to determine an approximation of the ergodic set of states without simulation, which can be sampled accordingly. However, as previously stated, we consider the approximation on simulated states an essential feature of our algorithm and, hence, it would be desirable to find sufficient conditions for convergence.

Furthermore, in some cases, we found that several KKT multipliers were "stuck" very close to zero for many episodes of training despite a positive value yielding much smaller errors in the equilibrium conditions. Recall that we use a softplus activation function in the output layer to ensure that the non-negativity constraints on the KKT multipliers are always satisfied.[31] A downside of using a softplus activation function is that its derivative decreases exponentially as its output approaches zero. Consequently, it can take a long time for small gradients to accumulate and lead to a larger output. If this problem occurs, a possible solution may be to use other (less standard) non-negative activation functions in the output layer,[32] or to use a different strategy to encode the KKT conditions into the loss function.[33]

[31] The softplus activation function is given by $\sigma(x) = \ln(1 + e^x)$.

[32] For example, $\sigma(x) = x^2$.

[33] Kotlikoff et al. (2020a) and Maliar et al. (2019), for example, make use of the Fischer-Burmeister function to replace the KKT conditions.

D.2 Passing redundant information to the equilibrium net

Heuristically, we found it helpful to pass redundant information to the neural network to stabilize the learning process. Since it increases the dimensionality of the input vector considerably, this is typically not done for methods like sparse grids (Krueger and Kubler, 2004), adaptive sparse grids (Brumm and Scheidegger, 2017), or Gaussian processes (Scheidegger and Bilionis, 2019). In our case, however, we found it helpful to increase the dimension of the input space by providing redundant information. Concretely, for the example studied in section 5, we augmented the input state with the TFP shock $\eta(z)$, the depreciation $\delta(z)$, the aggregate capital K and labor L, the return on capital r and the wage for labor w, the aggregate production Y, the distribution of financial wealth across age-groups $\mathbf{i}_{\text{fin}} = r \cdot \mathbf{k} + \mathbf{b} \cdot \min\{\kappa r, 1\}$, the distribution of labor income $\mathbf{i}_{\text{labor}} = w \cdot \mathbf{l}(z)$, the distribution of total wealth $\mathbf{i}_{\text{total}} = \mathbf{i}_{\text{labor}} + \mathbf{i}_{\text{fin}}$, and the transition probabilities of the exogenous shock $\Pi[z, :]$. To summarize, instead of passing the sufficient $2N + 1$ dimensional vector $\mathbf{x} = [z,\ \mathbf{k}^T,\ \mathbf{b}^T]^T$ to the neural network, we pass a $4N + 12$ dimensional vector $\mathbf{x}^* = [z,\ \eta(z),\ \delta(z),\ K,\ L,\ r,\ w,\ Y,\ \mathbf{k}^T,\ \mathbf{i}_{\text{fin}}^T,\ \mathbf{i}_{\text{labor}}^T,\ \mathbf{i}_{\text{total}}^T,\ \Pi[z, 1:4]]^T$ with redundant information to the neural network.[34]

D.3 Dealing with infeasible predictions at the beginning of training

At the beginning of the training process, we initialize the parameters of the neural network with random values. Consequently, the approximated policy, which predicts random values, might predict savings that imply that some age-groups consume negative amounts or that result in negative aggregate capital. Both of which cause the program to terminate with an error message. To deal with this issue, we replace infeasible values by feasible ones and add a punishment term to the loss function that trains the neural network not to predict policies that imply infeasible values. As learning proceeds and the approximate policies improve, these replacements are no longer necessary. Furthermore, it is important to verify that these replacements only occur at the beginning of the training process and never during simulation, after the training is finished.

[34] Note that $\mathbf{b}$ is implicitly passed to the neural network, because it is given by $\mathbf{b} = \frac{\mathbf{i}_{\text{fin}} - r \cdot \mathbf{k}}{\min\{\kappa r, 1\}}$.

The following method is rather *ad hoc*, however, we found it to work sufficiently well:

- Set a punishment parameter: $\epsilon^{\text{punish}} = 10^{-5}$.
- If the predicted consumption is less than ϵ^{punish}, replace by ϵ^{punish}: $\hat{c}^i_{\text{adjusted}}(\mathbf{x}) = \max(\hat{c}^i(\mathbf{x}), \epsilon^{\text{punish}})$.
- Add the punishment term $\left(\frac{1}{\epsilon^{\text{punish}}} \max(-\hat{c}^i(\mathbf{x}), 0)\right)^2$ to the loss function.

That way, if the neural network predicts a policy implying negative consumption, the optimality conditions can still be evaluated. At the same time, the large punishment term will guide the parameter updates so that improved policies do not imply negative consumption.

E An example with an analytical solution

In this section, we evaluate the performance of our solution method in a setting in which an exact solution is known. The knowledge of an exact solution allows us to evaluate the precision of our approximate solution beyond the relative errors in the Euler equations, which we have used so far. Furthermore, we use this example to demonstrate that our method is applicable in settings in which approximate aggregation may not hold.

We chose the same model as Krueger and Kubler (2004), which is an adapted version of the model in Huffman (1987). While analytically solvable, this model is closely related to those we solve in this paper.

E.1 The model

Next, for convenience and completeness, we outline this analytical test case. For readability, our notation here omits the dependence of variables on the node of the event tree, z^t, and we simply index them with time t, when there is no risk of confusion.

E.1.1 Households

Agents live for N periods and have log utility. Following Krueger and Kubler (2004) we assume that agents only work in the first period of their life: $l^s_t = 1$ for $s = 1$ and $l^s_t = 0$ otherwise. This implies that the aggregate labor supply is constant and equal to one: $L_t = 1$. Agents receive a competitive wage and can

save in risky capital. Households can not die with debt. Otherwise, there are no constraints or adjustment costs. The household's problem is given by

$$\max_{\{c^i_{t+i},a^i_{t+i}\}_{i=0}^{N-1}} \mathrm{E}_t\left[\sum_{i=0}^{N-1}\log(c^i_{t+i})\right] \tag{70}$$

subject to:

$$c^h_t + a^h_t = r_t k^h_t + l^h_t w_t \tag{71}$$

$$k^{h+1}_{t+1} = a^h_t \tag{72}$$

$$a^N_t \geq 0, \tag{73}$$

where c^h_t denotes the consumption of age-group h at time t, a^h_t denotes the saving, k^h_t denotes the available capital in the beginning of the period, and r_t denotes the price of capital.

E.1.2 Firms

There is a single representative firm with Cobb-Douglas production function. The production function is given by

$$f(K_t, L_t, z_t) = \eta_t K^\alpha_t L^{1-\alpha}_t + K_t(1-\delta_t), \tag{74}$$

where K_t denotes aggregate capital, L_t denotes the aggregate labor supply, α denotes the capital share in production, η_t denotes the stochastic TFP, and δ_t denotes the stochastic depreciation. The firm's optimization problem implies that the return on capital and the wage are given by $r_t = \alpha\eta_t K^{\alpha-1}_t L^{1-\alpha}_t + (1-\delta_t)$ and $w_t = (1-\alpha)\eta_t K^\alpha_t L^{-\alpha}_t$. Here we study the special case where $L_t = l^1_t = 1$.

E.2 Equilibrium

For completeness and convenience, we define the equilibrium we want to compute.

E.2.1 Competitive equilibrium

Definition 4 (competitive equilibrium) *A competitive equilibrium, given initial conditions $z_0, \{k^s_0\}_{s=1}^N$, is a collection of choices for households $\{(c^s_t, a^s_t)_{s=1}^N\}_{t=0}^\infty$ and for the representative firm $(K_t, L_t)_{t=0}^\infty$ as well as prices $(r_t, w_t)_{t=0}^\infty$, such that*

1. *given prices, households optimize,*

2. *given prices, the firm maximizes profits,*

3. *markets clear.*

E.2.2 Functional rational expectations equilibrium

Here, a FREE is defined as follows:

Definition 5 (functional rational expectations equilibrium) *A FREE consists of equilibrium functions* $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$*, where* $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ *denotes the capital investment functions, such that for all states* $\mathbf{x} := [z, \mathbf{k}^T]^T \in \mathcal{Z} \times \mathbb{R}^N$*, where* $z \in \mathcal{Z}$ *denotes the exogenous shock and* $\mathbf{k} = [k_1, \ldots, k_N]^T$ *denotes the endogenous state (*i.e.*, the distribution of capital) with* $k_1 = 0$*, we have for all* $i = 1, \ldots N-1$*:*

$$u'(c^i(\mathbf{x})) = \beta E_z \left[r(\mathbf{x}_+) u'(c^{i+1}(\mathbf{x}_+)) \right], \tag{75}$$

where $\mathbf{x}_+ = [z_+, 0, \boldsymbol{\theta}_a(\mathbf{x})^T]^T$*, where* z_+ *denotes the random exogenous shock in the next period, and where*

$$r(\mathbf{x}) = f_K \left(\sum_{i=1}^{N} x_{1+i}, 1, x_1 \right), \tag{76}$$

$$w(\mathbf{x}) = f_L \left(\sum_{i=1}^{N} x_{1+i}, 1, x_1 \right), \tag{77}$$

$$c^i(\mathbf{x}) = \begin{cases} w(\mathbf{x}) - \theta_a(\mathbf{x})_i, & \text{for } i = 1, \\ r(\mathbf{x}) x_{1+i} - \theta_a(\mathbf{x})_i, & \text{for } i = 2, \ldots, N-1, \\ r(\mathbf{x}) x_{1+N} & \text{for } i = N. \end{cases} \tag{78}$$

E.3 Computing the equilibrium

We chose to present this model because it is closely related to the models we study in this paper while an exact solution is available. In this section, we provide the exact solution and briefly reiterate how the equilibrium would be approximated using deep equilibrium nets.

E.3.1 Exact solution

Following Krueger and Kubler (2004), a functional rational expectations equilibrium is given by

$$\boldsymbol{\theta}_a(\mathbf{x}) = \beta \left[\frac{1-\beta^{N-1}}{1-\beta^N} w(\mathbf{x}), \frac{1-\beta^{N-2}}{1-\beta^{N-1}} r(\mathbf{x}) k_2, \frac{1-\beta^{N-3}}{1-\beta^{N-2}} r(\mathbf{x}) k_3, \ldots, \frac{1-\beta^1}{1-\beta^2} r(\mathbf{x}) k_{N-1} \right]^T. \tag{79}$$

This solution implies that aggregate capital evolves according to

$$K_t = \gamma_1 w(\mathbf{x}_{t-1}) + \gamma_2 w(\mathbf{x}_{t-2}) r(\mathbf{x}_{t-1}) + \cdots + \gamma_{N-1} w(\mathbf{x}_{t-(N-1)}) r(\mathbf{x}_{t-1}) r(\mathbf{x}_{t-2}) \ldots r(\mathbf{x}_{t-(N-2)}), \tag{80}$$

where $\gamma_i := (\beta^i - \beta^N)/(1-\beta^N)$.

E.3.2 Approximating the solution with deep equilibrium nets

We seek to find parameters $\boldsymbol{\rho}$ of a deep neural network $\mathcal{N}_{\boldsymbol{\rho}}$, such that it approximates the true equilibrium policies:

$$\hat{\boldsymbol{\theta}}_a(\mathbf{x}) := \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) \approx \boldsymbol{\theta}_a(\mathbf{x}). \tag{81}$$

Since, in general, the true policy $\boldsymbol{\theta}_a(\mathbf{x})$ is unknown, we follow the algorithm described in this paper and find parameters $\boldsymbol{\rho}$ by minimizing the errors in the equilibrium conditions with variants of mini-batch gradient descent. We can plug market clearing and the firms' optimization into the Euler equations so that the Euler equations characterize the recursive equilibrium. Given neural network parameters $\boldsymbol{\rho}$, which imply approximate policy functions $\hat{\boldsymbol{\theta}}_a(\mathbf{x})$, the errors in the equilibrium conditions at state $\mathbf{x}_j$ are given by

$$e^i_{\text{EE}}(\mathbf{x}_j) := -u'(\hat{c}^i(\mathbf{x}_j)) + \beta \mathrm{E}_z \left[r(\hat{\mathbf{x}}_+) u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_+)) \right], \text{ for } i = 1 \ldots N-1, \tag{82}$$

where

$$\hat{\mathbf{x}}_+ = \left[z_+, \ 0, \ \hat{\boldsymbol{\theta}}_a(\mathbf{x})^T \right]^T, \tag{83}$$

and where z_+ denotes the random exogenous shock in the next period, and

$$r(\mathbf{x}) = f_K\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{84}$$

$$w(\mathbf{x}) = f_L\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{85}$$

$$\hat{c}^i(\mathbf{x}) = \begin{cases} w(\mathbf{x}) - \hat{\theta}_a(\mathbf{x})_i, & \text{for } i = 1, \\ r(\mathbf{x})x_{1+i} - \hat{\theta}_a(\mathbf{x})_i, & \text{for } i = 2, \ldots, N-1, \\ r(\mathbf{x})x_{1+N} & \text{for } i = N. \end{cases} \tag{86}$$

The relative errors in the Euler equations are given by

$$e^i_{\text{REE}}(\mathbf{x}_j) := \frac{u'^{-1}\left(\beta \mathrm{E}_{z_j}\left[r(\hat{\mathbf{x}}_{j,+})u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+}))\right]\right)}{\hat{c}^i(\mathbf{x}_j)} - 1 \tag{87}$$

and the loss function for the deep equilibrium net is given by

$$\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|}\frac{1}{N-1}\sum_{\mathbf{x}_j \in \mathcal{D}_{\text{train}}}\sum_{i=1}^{N-1} e^i_{\text{REE}}(\mathbf{x}_j)^2, \tag{88}$$

where $\mathcal{D}_{\text{train}}$ denotes a set of states collected by simulating the economy using the current approximation of the policy functions.

E.4 Parameterization and hyper-parameters

We solve the model for $N = 6$ agents and base our parameterization on Krueger and Kubler (2004). The chosen economic parameters as well as the chosen hyper-parameters are given in table 10.

E.5 Assessing the quality of the solution

We train the neural network for a total of $10,000$ episodes. We first evaluate the quality of the obtained policy functions by looking at the relative errors in the Euler equations, since this is the measure we focus on in the settings we are interested in and where no exact solution is available. Table 11 shows the relative errors in the Euler equations on $15,000$ randomly simulated states.[35] The mean errors are in the order of $\sim 10^{-4}$, while the 99.9th percentile is below

[35] We simulate $16,000$ periods and throw away the first 1000.

Number age-groups N	Discount factor β	Relative risk aversion γ	Capital share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1} = 1.05 \vert \eta_t = 1.05)$ $P(\eta_{t+1} = 0.95 \vert \eta_t = 0.95)$	Persistence depreciation $P(\delta_{t+1} = 0.5 \vert \delta_t = 0.5)$ $P(\delta_{t+1} = 0.9 \vert \delta_t = 0.9)$
6	0.7	1	0.3	$\{0.95, 1.05\}$	$\{0.5, 0.9\}$	0.5 0.5	0.5 0.5

Learning rate α^{learn}	Periods per episode T	Epochs per episode N^{epochs}	Mini-batch size m	Nodes hidden layers	Activations hidden layers
0.00001	12,800	20	640	100 50	relu relu

Table 10: Parameterization of the economic parameters (top) and the hyper-parameters of the neural network (bottom).

	mean	max	0.1	10	50	90	99.9
Rel Ee capital [log10]	-3.4	-2.4	-6.4	-4.4	-3.6	-3.0	-2.5

Table 11: Relative Euler equation errors on 15,000 simulated periods. The columns show the mean and the max errors as well as the 0.1st, 10th, 50th, 90th, and 99.9th percentile.

1%. Since this is a much simpler model than our benchmark model, the errors are slightly lower than the errors we obtained for the models discussed in the previous sections.

Since the exact solution is available in this model, we now compare the approximate policy learned by our method to the precise solution. Figure 11 shows the savings decisions learned by the deep equilibrium net (shown as stars) as well as the precise solution (shown as circles) as a function of the agents' capital at the beginning of each period in a scatter plot. The figure shows only 200 simulated states for better readability. The colors indicate the four exogenous shocks. Visually, the policies are indistinguishable. Table 12 shows statistics of the errors in the learned savings decisions. The mean errors in the policy functions are of order $\sim 10^{-4}$ and the maximum errors stay below 0.2%. In the next section, we study the extent to which an accumulation of errors could influence aggregates in this model.

E.6 Approximate aggregation

This section serves two purposes. First, we want a measure of the accumulation of errors with time as we simulate the economy forward. Second, we want to see to what extent approximate aggregation, as in the seminal work of Krusell

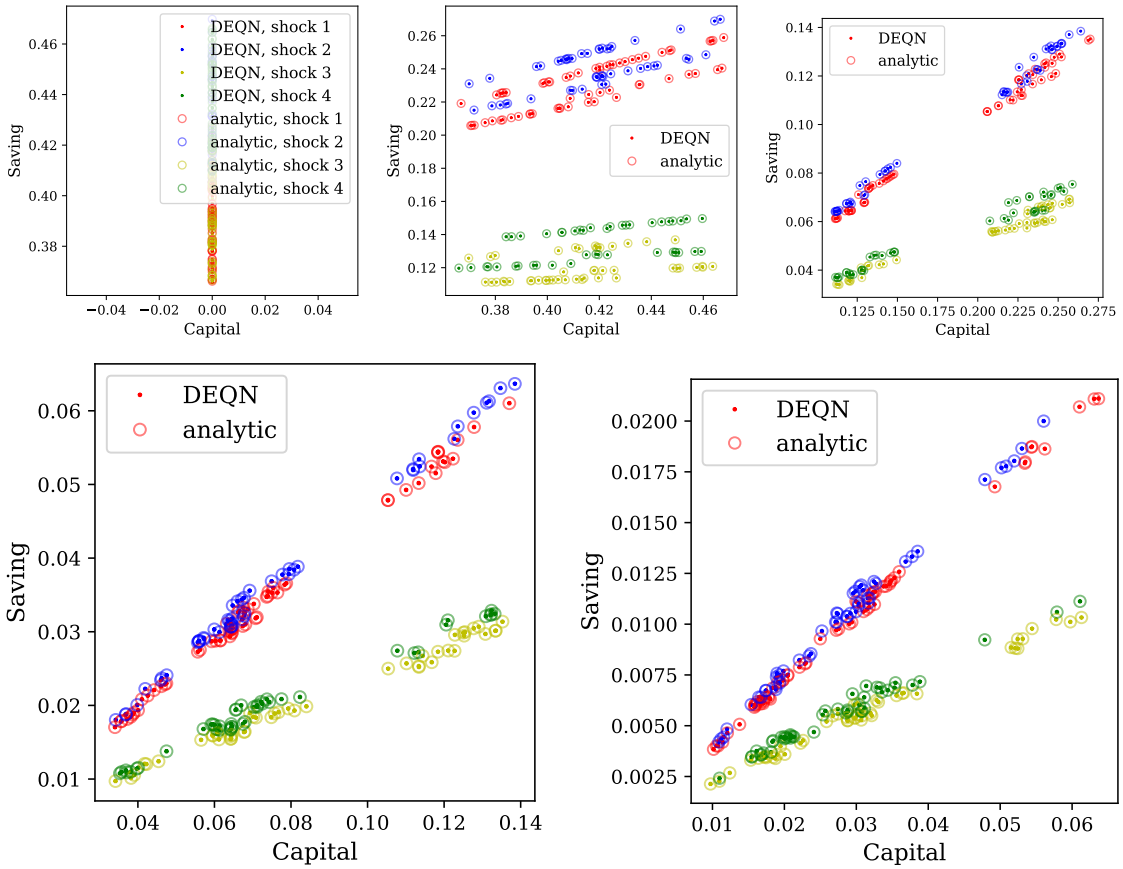


Figure 11: Exact and approximate saving decisions plotted against capital holding at the beginning of the period for 200 simulated states for each of the 6 age-groups. The circles show the exact solution, and the stars show the solutions learned by the deep equilibrium net after training for 10,000 episodes with parameters shown in Table 10. The first row shows the policies for age-groups 1, 2, and 3, the second row shows the policies for age-groups 4 an 5. The policy of the 6th age-group is trivial (consume everything and save nothing). Shock 1 is denoted in red, shock 2 in blue, shock 3 in yellow, and shock 4 in green. The TFP and deprecation $(\eta,\ \delta)$ of shock 1, shock 2, shock 3, and shock 4 are $(0.95,\ 0.5)$, $(1.05,\ 0.5)$, $(0.95,\ 0.9)$, and $(1.05,\ 0.9)$, respectively.

	mean	max	0.1	50	99.9
Relative policy errors age-group 1 [%]	0.03	0.14	0.00	0.03	0.13
Relative policy errors age-group 2 [%]	0.02	0.09	0.00	0.01	0.08
Relative policy errors age-group 3 [%]	0.02	0.10	0.00	0.02	0.09
Relative policy errors age-group 4 [%]	0.01	0.05	0.00	0.01	0.14
Relative policy errors age-group 5 [%]	0.01	0.06	0.00	0.01	0.04
Relative policy errors age-group 1 [log10]	-3.47	-2.85	-6.08	-3.54	-2.88
Relative policy errors age-group 2 [log10]	-3.82	-3.06	-6.72	-3.91	-3.10
Relative policy errors age-group 3 [log10]	-3.69	-2.99	-6.40	-3.75	-3.04
Relative policy errors age-group 4 [log10]	-4.09	-3.29	-7.11	-4.26	-3.37
Relative policy errors age-group 5 [log10]	-3.92	-3.33	-6.70	-4.00	-3.35

Table 12: Statistics of the relative errors in the learned policy functions on 15,000 simulated periods. The columns show the mean and the max errors as well as the 0.1, 50 and 99.9th percentile.

	mean	max
DEQN [%]	0.019	0.13
Log-linear forecast [%]	0.24	1.3
DEQN [log10]	-3.72	-2.88
Log-linear forecast [log10]	-2.62	-1.87

Table 13: Statistics of the relative errors in the sequence of aggregate capital compared to the exact solution when simulating 15,000 time periods.

and Smith (1998), holds in this model.

Following Krueger and Kubler (2004), we only use the log of aggregate capital to forecast next-period's capital contingent on the exogenous shock when assessing the approximate aggregation in this model.[36] More precisely, the functional form of the forecasting rule is given by

$$\log(K_{t+1}) = \phi_{0,z} + \phi_{1,z} \log(K_t), \tag{89}$$

where we estimate the parameters $\phi_{0,z}$ and $\phi_{1,z}$ with an ordinary least squares regression for each of the four shocks. The R^2 values of the corresponding regressions are between 0.9978 and 0.9979. The corresponding mean forecasting errors are between 0.16% and 0.24%. The corresponding maximum forecasting errors are between 0.61% and 0.98%.

Following Den Haan (2010) and Winberry (2018), we also analyze the extent to which approximate aggregation holds by simulating the linear forecast of aggregate capital forward without updating the value of aggregate capital. We do the same using the solution learned by the deep equilibrium net and compare the sequences of aggregate capital on $15,000$ simulated states. Table 13 reports the mean and maximum of the relative error of the sequence of aggregate capital compared to the exact solution. The mean relative errors in the sequence of aggregate capital are more than one order of magnitude lower using the policies learned by the deep equilibrium net than when using a log-linear approximation of the law of motion. To illustrate this point figure 12 shows the simulated values of aggregate capital for the exact solution (red circles), using the policies learned by the deep equilibrium nets (blue dots), and when using a log-linear forecast (green crosses). For readability, we show the first 100 and the last 1000 periods. The figure shows that the evolution of aggregate capital obtained from the policies learned by the deep equilibrium net is visually indistinguishable from the exact solution, while the evolution using a

[36] The regressions use 15,000 simulated states in total, approximately 3750 states for each shock.

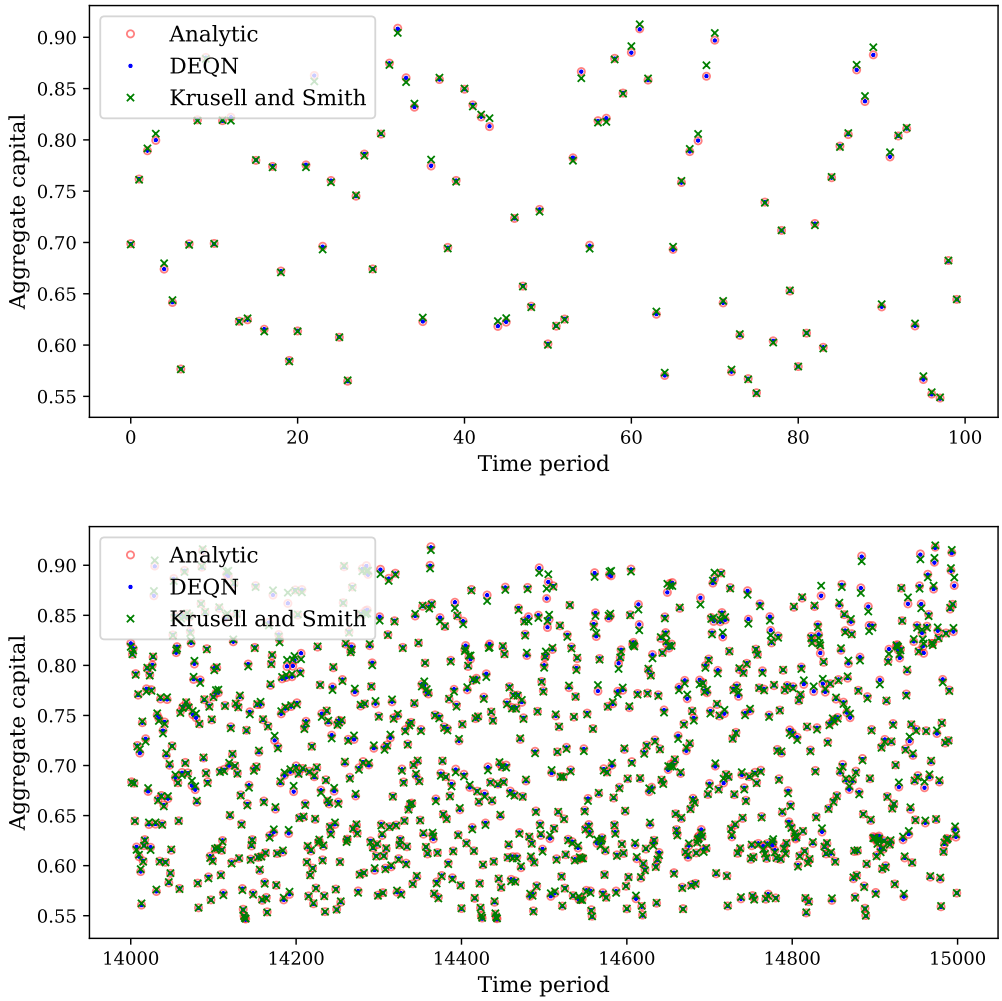


Figure 12: Exact and approximated evolution of aggregate capital. The red circles show the exact evolution, the blue stars show the evolution implied by the policies learned by the deep equilibrium net, and the green crosses show the evolution when using a log-linear forecast. The upper figure shows the first 100 and the lower figure the last 1000 periods of 15,000 simulated periods.

log-linear forecast shows visible differences. This further illustrates the point that our method applies to settings in which approximate aggregation does not obtain. For both approximate solutions, we find no evidence that the quality of the solution deteriorates when simulating many periods.

As noted in Krueger and Kubler (2004), in this simple model, errors in forecasting the aggregate capital stock do not translate to welfare losses for the agents. This is because, in the simple model presented here, the optimal decision does not depend on interest rates.

F Parallelization scheme

To speed up the solution process of the computational method introduced in this paper by orders of magnitude, we present a generic and highly-scalable parallelization scheme that allows us to make efficient use of state-of-the-art high-performance computing (HPC) facilities, whose performance nowadays can reach up to hundreds of Petaflop/s (see, *e.g.*, https://www.top500.org). The use of modern HPC hardware may enable us to solve hard problems with many state variables in a relatively short time and to tackle problems that have, thus far, been out of reach.

Our algorithm can be described as iterating on two phases: the simulation phase, where new training data is generated using the previous iteration's neural network, and the learning phase, where the neural network is trained on the new data. We make use of contemporary frameworks to parallelize both phases. More precisely, our method is parallelized by combining TensorFlow (Abadi et al., 2015) with Horovod (Sergeev and Del Balso, 2018).

Horovod is an open-source distributed training framework for TensorFlow (and other deep learning codes) that was developed to make distributed deep learning fast and easy to use. The core idea behind distributed deep learning is the following: first, training data is distributed across the compute nodes in a cluster. On each node, the gradient of the cost function with respect to the parameters of the neural network on its given batch of data is determined. Finally, all nodes are synchronized such that the gradient descent step can be taken using the average gradient across the nodes. Horovod implements the synchronization and calculation of the average gradient using a Ring Allreduce

operation.[37,38]

Additionally, our simulation phase also lends itself nicely to parallelism: each node simulates the part of the training data, which it will then use to compute the gradient during the learning phase. This also limits the communication between the nodes. The only communication required among the nodes is the communication of the gradients for the Ring Allreduce operation. In section F.1, we provide additional details on the parallelization and in section F.2, we demonstrate that this parallelization scheme can accelerate the computations by orders of magnitude.

F.1 Hybrid parallelization scheme for heterogeneous HPC systems

In the following, we detail the hybrid parallelization scheme, which is schematically displayed in figure 13. Let N_N denote the number of nodes available in the computing system and $n \in \{1, \ldots, N_N\}$ denote a specific node. Each node has a copy of the same neural network with identical weights, such that $\boldsymbol{\rho}_i = \boldsymbol{\rho}_j$, $\forall i, j \in \{1, ..., N_N\}$. Then, each node has a different seed for random number generators to ensure that each node generates different data during the simulation phase. The following summarizes the parallelization scheme for a given episode i.

Simulation phase

On each node n:

- Draw a random sequence of exogenous shocks and simulate $\tilde{T} := T/N_N$ periods using the policy encoded by the neural network $\mathcal{N}_{\boldsymbol{\rho}_n}$.
- Create $N_m := \tilde{T}/m$ random mini-batches. We denote the mini-batches as $b_{i,n,j}$ and the set of mini-batches by $\mathcal{B}_{i,n} = \cup_{j=1}^{N_m} b_{i,n,j}$.

Learning phase

On each node n:

For each mini-batch $b_{i,n,j} \in \mathcal{B}_{i,n}$ (*i.e.*, for $j = 1, \ldots, N_m$):

[37] The Horovod source code was based on the Baidu TensorFlow-allreduce repository written by Andrew Gibiansky and Joel Hestness (*cf.* https://github.com/baidu-research/tensorflow-allreduce).

[38] Ring Allreduce is an efficient version of the Allreduce operator commonly used in HPC. Ring Allreduce mitigates a bottleneck in the amount of data sent between nodes. Allreduce is tied to another operator, in our case, the mean. At the end of Ring Allreduce, each node has a copy of the Allreduced data.

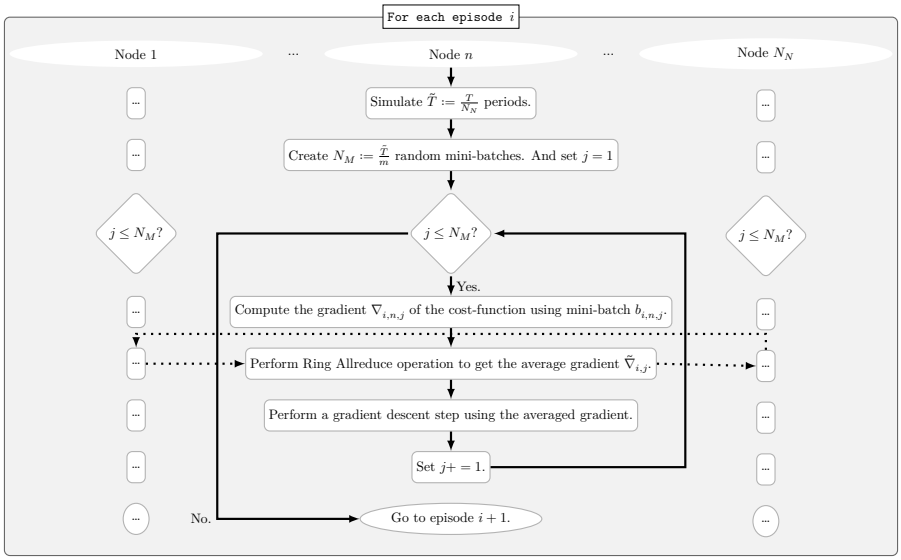


Figure 13: Schematic illustration of the parallelization scheme.

- Compute the gradient of the cost function with respect to the parameters of the neural network $\boldsymbol{\rho}_n$, denoted as $\nabla_{i,n,j}$.
- Perform a Ring Allreduce operation to obtain the average gradient across nodes,

$$\tilde{\nabla}_{i,j} = \frac{1}{N_N} \sum_{k=1}^{N_N} \nabla_{i,k,j}.$$

- Update the parameters of the neural network doing a gradient descent step as implied by the average gradient ($\tilde{\nabla}_{i,j}$). New weights: $\boldsymbol{\rho}_n^{\text{new}} = \boldsymbol{\rho}_n^{\text{old}} - \alpha^{\text{learn}} \tilde{\nabla}_{i,j}$.

F.2 Strong scaling

In this section, we report the strong scaling and the efficiency of our code on a simple benchmark problem, similar to the model studied in 5. This scaling test was carried out on the Cray XC50 "Piz Daint" system that is installed at the Swiss National Supercomputer Centre (CSCS). Cray XC50 compute nodes combine Intel Xeon E5-2690 v3 CPUs (12 cores, 64GB RAM) with one NVIDIA P100 GPU that is equipped with 16GB of memory.

In our strong scaling example, we chose the network architecture to be 1500 nodes for the two hidden layers, a batch-size of 150, 000, and an episode length of 76, 800, 000 periods. These numbers are larger than those used above,

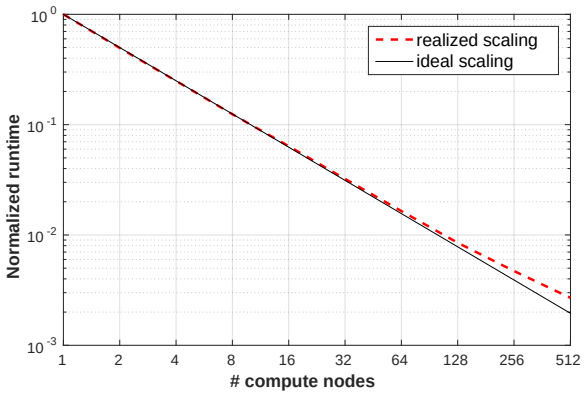


Figure 14: Strong scaling.

reflecting the fact that we aim at using larger networks and batch-sizes to study more complicated models in the future. Figure 14 shows the resulting strong scaling and efficiency. The runtime is normalized by the time it takes a single node to simulate and train on one episode. As displayed, we find excellent strong scaling properties, with a parallel efficiency above 75%, even at 512 hybrid compute nodes.