Deep equilibrium nets*

Marlon Azinović, Luca Gaegauf, Simon Scheidegger†

May 24, 2019

Abstract

In order to develop large-scale overlapping generations models, which are consistent with findings at the micro-level, one has to be able to include a substantial amount of heterogeneity, significant uncertainty, and financial frictions. Studying such models demands that one can compute equilibria in situations jointly featuring a high-dimensional state space, kinks in the equilibrium functions, and irregular state space geometries. To this end, we introduce deep equilibrium nets—neural networks that directly approximate all equilibrium functions in discrete-time dynamic stochastic economic models 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, 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. To demonstrate the performance of the proposed method, we study the effects of borrowing constraints and adjustment costs on the cross-sectional consumption response to aggregate shocks in overlapping generations models with 60 generations, aggregate uncertainty, a one-period bond, and occasionally binding constraints. We obtain average relative errors in the Euler equations of the order $\sim 10^{-4}$ when applying a densely connected deep neural network with two hidden layers. To study the effect of borrowing constraints on the cross-sectional consumption response to stochastic shocks to total factor productivity and depreciation of capital, we compare two economies that differ in the level of the exogenous borrowing constraint. In one economy, agents are not allowed to take up debt; in the other, agents can take up debt up to an exogenously fixed value that we set to -84% of mean yearly per-capita consumption. The effect of looser borrowing constraints differs across age-groups: the consumption response to a shock with high total factor productivity and high depreciation, for example, decreases from 14.4% to 4.1% for 21-year-old agents, while it increases from 12.6% to 21.6% for 32-year-old agents and decreases from 4.1% to 3.6% in the aggregate.

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 suggestions by Karl Schmedders and participants at the *Advances in Computational Economics and Finance* Seminar and the *Brown bag lunch seminar* at the University of Zurich. This work was generously supported by grants from the Swiss National Supercomputing Centre (CSCS) under project ID s885 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 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

The rising wealth and earnings inequality (see, *e.g.*, Krueger et al., 2010; Saez and Zucman, 2016; Zucman, 2019) as well as the financial crisis and its aftermath, call for micro-founded macroeconomic models. To answer this call, heterogeneity between types of agents, such as hand-to-mouth and non hand-to-mouth consumers (see, *e.g.*, 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)) have become widely recognized as key ingredients for modern macroeconomic models. However, despite recent advances in the field of computational economics (see, *e.g.*, Judd et al., 2011; Maliar and Maliar, 2015; Brumm and Scheidegger, 2017; Cai et al., 2017; Scheidegger and Bilionis, 2019; Kubler and Scheidegger, 2018; Duarte, 2018a; Villa and Valaitis, 2018; Fernández-Villaverde et al., 2019), there is still no tractable computational method that is capable of solving economic models in discrete-time, jointly featuring 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. This is because the curse of dimensionality (Bellman, 1961), which arises from 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 which are tailored to approximating such functions, such as adaptive sparse grids (see, *e.g.*, Brumm and Scheidegger, 2017; Brumm et al., 2017; Scheidegger and Treccani, 2018), fail if the dimensionality of the state space exceeds ~ 20. While there are methods available that can handle a subset of a), b), c), and d), there exists, to the best of our knowledge, at present, no tractable method that can deal with dynamic models that feature all four. Adaptive sparse grids, for example, only address challenges a), b), and c), while Gaussian processes (see Renner and Scheidegger, 2018; Scheidegger and Bilionis, 2019) only address a), b), and d).

In this paper, we present a novel algorithm, which is based on unsupervised machine learning, that leverages recent advances in deep learning (see, *e.g.*, Goodfellow et al., 2016, for a general introduction) to compute approximate recursive equilibria in discrete-time models with a) stochasticity, b) a very 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[1] on simulated data. We refer to our neural network, which approximates all equilibrium functions directly as *deep equilibrium net*. The key idea of our proposed (unsupervised machine learning) 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.[2] The states used to train the deep equilibrium net are sampled by simulating the economy; thereby, the network learns to approximate the equilibrium only where it matters, *i.e.*, on an approximation of the ergodic set. In contrast to most grid based methods, such as standard value function or policy function iteration with regular or sparse grids (see, Krueger and Kubler, 2004; Judd et al., 2014; Brumm and Scheidegger, 2017; Brumm et al., 2017, for Smolyak sparse grids and adaptive sparse grids), our method is thus able to approximate the equilibrium on irregularly shaped geometries, which is an advantage, especially in high dimensions (see also Judd et al., 2011; Maliar and Maliar, 2015; Scheidegger and Bilionis, 2019, for computational methods that also address this issue).

[1] We use Adam, introduced by Kingma and Ba (2014), which is considered the de facto standard, at present, in the field of deep learning. See appendix A.1 for details.

[2] For a general introduction to machine learning, see, *e.g.*, Murphy (2012).

The main contributions of this paper are fourfold: first, we introduce a grid-free, generic method based on simulations and deep neural networks to compute global solutions[3] to high-dimensional, and potentially highly non-linear, dynamic stochastic models. Second, the formulation of our method 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 very rich models that may be a better description of the real world than the ones that have been used by practitioners to date. Fourth, to illustrate the capabilities of our proposed algorithm, we solve a classic overlapping generation model from the literature but extend it substantially by adding borrowing constraints to the model. We find that the effect of borrowing constraints on the consumption response to aggregate shocks is heterogeneous across age-groups. Important consequences of tighter or looser borrowing constraints are hence masked when only looking at aggregates, underlining the importance of modeling the life-cycle in models with aggregate uncertainty and financial frictions. Next, we further enrich the model by adding convex adjustment costs to investments in capital and, additionally, allowing age-groups to trade a one-period bond subject to collateral constraints.

We illustrate the capabilities of our novel method by solving an annually calibrated life-cycle model with agents that live for 60 periods, borrowing constraints, and aggregate uncertainty, which amounts to about 120 dimensions. We chose this type of model for two 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 agents' life-cycle is of crucial importance to study. for instance, social security (see, *e.g.*, Krueger and Kubler, 2006), climate change (see, Kotlikoff et al., 2019), 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, 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 (see, *e.g.*, Press et al., 2007), quickly become infeasible with an increasing size of the state space due to the curse of dimensionality (see Bellman, 1961). The curse of dimensionality—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; Judd et al., 2014) or adaptive sparse grids (see Brumm and Scheidegger, 2017; Brumm et al., 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 of irregular shape, it often fills out only a tiny fraction of the hypercube (see Judd et al., 2011; Maliar and Maliar, 2015; Scheidegger and Bilionis, 2019 for details). Thus, approximating the economy on a hypercubic domain can be wasteful and, in some cases, render the computation of solutions

[3] We follow Brumm and Scheidegger (2017) and use the term "global solution" for a solution that is computed using equilibrium conditions at many points in the state space of a dynamic model—in contrast to a "local solution", which rests on a local approximation around a steady state of the model. For a method that computes such a global solution, we use the term "global solution method". This use of the word "global" is not to be confused with its use in the phrase "global optimization method", which refers to a method that aims to find a global optimum.

[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).

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, *e.g.*, Scheidegger and Bilionis, 2019; Renner and Scheidegger, 2018; Kubler and Scheidegger, 2018). Gaussian processes, however, are not capable of dealing with a large amount of input data[5] (see Rasmussen, 2004) and hence can not exploit the fact that simulation-based methods 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 (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), they are a grid-free method, and they excel when there is much data[6] 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 economic application. Section 4 introduces our proposed solution method. In section 5, we showcase our novel solution method in the context of a large-scale economic model in great detail. Thereafter, section 6 illustrates the economic insights that can be obtained when solving life-cycle models with agents living for 60 periods, aggregate uncertainty, and borrowing constraints. Section 7 presents the application of the proposed method in a model with illiquid, risky capital, a one-period bond, and collateral constraints. Section 8 concludes.

2 Literature review

The method we propose in this paper substantially contributes to three particular strands of literature on computing recursive equilibria numerically (see fig. 1 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 a high-dimensional state space, 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 can consequently approximate equilibria more efficiently[7] 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 compute approximate equilibrium functions directly. To the best of our knowledge, we are the first to be doing 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 papers, which review existing computational methods, are available, this literature review focuses on placing our paper within the emerging strand of literature that

[5]Standard Gaussian processes become computationally intractable for more than $\sim 10^4$ observations, as their computational complexity scales as $\mathcal{O}(N^3)$, where N is the number of observations.

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

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

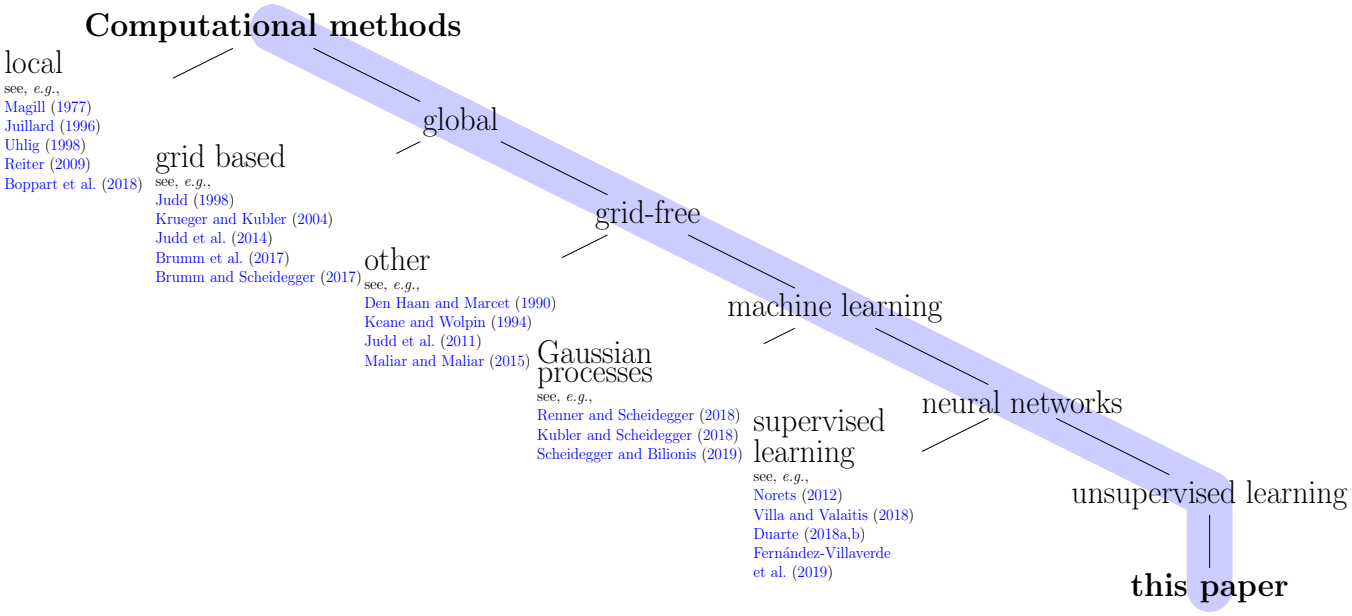


Figure 1: Illustration of the placement of this paper within the literature.

makes explicit use of machine learning to compute equilibria in economic models. Maliar and Maliar (2014) and Kollmann et al. (2011) give excellent overviews on computational methods for economic models with finitely many agents. For an overview on 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) 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 advantage 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. At the time of writing this paper, two machine learning methods have been favored in particular in economic applications: Gaussian process (GP) regression and neural networks.

GP regression is a Bayesian method: the prior beliefs about the response are encoded in the choice of the prior covariance functions. Thus, given the modelers' expert knowledge, they usually approximate the functions of interest with relatively few observations (Rasmussen, 2004; Scheidegger and Bilionis, 2019). However, as ordinary GPs are unable to deal with input dimensions larger than about 20 (Scheidegger and Bilionis, 2019), other dimension reductions methods have to be applied in combination. For example, Kubler and Scheidegger (2018) and Scheidegger and Bilionis (2019) achieve this by augmenting GPs by active subspaces.

Next to GPs, neural networks have also piqued the interest of computational economists. Neural networks are particularly promising since they have shown 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). An early application of neural networks in the field of computational economics is Norets (2012), who uses neural networks in the context of estimation and discrete state dynamic programming. He 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. The neural network is trained with supervised learning. Duarte (2018b,a) 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 elegantly and 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 experts' 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 make additional 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 (2018) 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. Similarly to our algorithm, their method iterates between a stochastic simulation phase and a learning phase. In contrast to Villa and Valaitis (2018), we do unsupervised learning.

3 The economic model

To illustrate the capabilities of our proposed method, we chose an economic model whose dimensionality can be scaled in a straight forward, but meaningful way. To this end, we use a model that closely follows the work of Krueger and Kubler (2004), namely, an overlapping generation (OLG) model with stochastic production in which the dimensionality of the state space increases linearly with the number of periods the agents live. However, and in contrast to Krueger and Kubler (2004), we consider an OLG model where the agents face borrowing constraints.

3.1 Uncertainty

The model we consider is in discrete time—that is, $t = 0, \ldots, \infty$. An event tree denotes uncertainty. The root of the tree is denoted by z_0. Each node of the event tree corresponds to a history of the aggregate shocks $z^t = (z_0, z_1, \ldots, z_t)$. The shocks are assumed to follow a Markov chain with finite support $\mathcal{Z}$ and with transition matrix Π.

3.2 Households

We study an overlapping generation model where agents live for N periods. There is no uncertainty about the lifetime. 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. Each period, the agents alive receive a strictly positive labor endowment which depends on the exogenous shock and the age of the agent alone. The labor endowment in period t of an agent of age s is denoted by $l^s(z_t) = l_t^s$. The price of the consumption good is normalized to one at each node z^t and the household supplies its 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 $\beta < 1$ is the discount factor, and the utility function $u : \mathbb{R}_{++} \to \mathbb{R}$ is assumed to be smooth, strictly increasing, strictly concave, and to satisfy the Inada condition $\lim_{c\to 0} u'(c) = \infty$. Households can save a unit of consumption good to obtain a unit of capital good next period. The 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:

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

Note that in Krueger and Kubler (2004), except that households can not die with debt, borrowing is not restricted. In our version of OLG model, however, we consider the case where borrowing is allowed up to an exogenously given level $\underline{a}$, *i.e.*,

$$a_t^s \geq \underline{a}. \tag{3}$$

At time t, the households sells its capital (k_t^s) to the firm at market price $r_t > 0$. The budget constraint of household s in period t is

$$c_t^s + a_t^s = r_t k_t^s + l_t^s w_t. \tag{4}$$

Finally, the agents are born and die without any assets (*i.e.*, $k_t^1 = 0$ and $a_t^N = 0$).

3.3 Firms

There is a single representative firm with 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

$$f(K, L, z) = \eta(z) K^\alpha L^{1-\alpha} + K(1 - \delta(z)), \tag{5}$$

where K denotes the aggregate capital bought, L denotes the aggregate labor hired, α denotes the capital share in productions, $\eta(z)$ denotes the stochastic TFP, and $\delta(z)$ denotes the stochastic depreciation.

3.4 Markets

In the formulation of the OLG model we are solving below, we assume that there are competitive spot markets for consumption, capital, and labor.

3.5 Equilibrium

Following Krueger and Kubler (2004), we define a competitive equilibrium for our economy:

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

1. *Given* $(r_t, w_t)_{t=0}^\infty$, *the choices* $\{(c_t^s, a_t^s)_{s=1}^N\}_{t=0}^\infty$ *maximize* (1), *subject to* (2), (3), *and* (4).
2. *Given* r_t, w_t, *the firm maximizes profits,* i.e.,

$$(K_t, L_t) \in \underset{K_t, L_t \geq 0}{\arg\max}\, f(K_t, L_t, z_t) - r_t K_t - w_t L_t. \tag{6}$$

3. *All markets clear: For all* t

$$L_t = \sum_{s=1}^N l_t^s, \tag{7}$$

$$K_t = \sum_{s=1}^N k_t^s, \tag{8}$$

For our choice of the production function (eq. (5)), the first order conditions of the firms maximization problem imply that

$$w(z^t) = (1-\alpha)\eta(z_t)K(z^t)^\alpha L(z^t)^{-\alpha}, \tag{9}$$

$$r(z^t) = \alpha\eta(z_t)K(z^t)^{\alpha-1}L(z^t)^{1-\alpha} + (1-\delta(z_t)). \tag{10}$$

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:

$$u'(c^s(z^t)) = \beta \mathrm{E}_{z_t}\left[u'(c^{s+1}(z^t, z_{t+1}))r(z^t, z_{t+1})\right] + \lambda^s(z^t), \tag{11}$$

$$\lambda^s(z^t) \cdot (a^s(z^t) - \underline{a}) = 0, \tag{12}$$

$$a^s(z^t) - \underline{a} \geq 0, \tag{13}$$

$$\lambda^s(z^t) \geq 0. \tag{14}$$

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

4 Approximation of equilibria with deep neural networks

We now describe how to solve the OLG model presented in section 3 via deep equilibrium nets. To this end, we proceed in two steps. In section 4.1, we define functional rational expectations equilibria for the presented economy. Section 4.2 describes how to use deep neural networks to search for approximate recursive equilibria by using projection methods (see, *e.g.*, Judd, 1992 and Gaspar and Judd, 1997). Note that while we present our novel method in the context of OLG models, it is generic by construction—that is, it could be applied to any other discrete-time dynamic stochastic model.

4.1 Functional rational expectations equilibrium

The algorithm we introduce below, in section 4.2.5, searches for a recursive equilibrium where the distribution of capital holdings constitutes a sufficient endogenous state. Following Spear (1988) and Krueger and Kubler (2004), we call this a *functional rational expectations equilibrium* (FREE). More precisely, we define a FREE as

Definition 2 (functional rational expectations equilibrium) *A FREE consists of equilibrium functions* $\boldsymbol{\theta} = [\boldsymbol{\theta}_a^T, \boldsymbol{\theta}_\lambda^T]^T : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{2(N-1)}$*, where* $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ *denotes the capital investment functions and* $\boldsymbol{\theta}_\lambda : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ *denotes the KKT multiplier 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* $i = 1, \ldots N-1$*:*

$$u'(c^i(\mathbf{x})) = \beta E_z \left[r(\mathbf{x}_+) u'(c^{i+1}(\mathbf{x}_+)) \right] + \theta_\lambda(\mathbf{x})_i, \tag{15}$$

$$\theta_\lambda(\mathbf{x})_i (\theta_a(\mathbf{x})_i - \underline{a}) = 0, \tag{16}$$

$$(\theta_a(\mathbf{x})_i - \underline{a}) \geq 0, \tag{17}$$

$$\theta_\lambda(\mathbf{x})_i \geq 0 \tag{18}$$

where

$$\mathbf{x}_+ = \begin{bmatrix} z_+ \\ 0 \\ \boldsymbol{\theta}_a(\mathbf{x}) \end{bmatrix}, \tag{19}$$

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

$$r(\mathbf{x}) = f_K \left(\sum_{i=1}^N x_{1+i}, \sum_{i=1}^N l^i(x_1), x_1 \right), \tag{20}$$

$$w(\mathbf{x}) = f_L \left(\sum_{i=1}^N x_{1+i}, \sum_{i=1}^N l^i(x_1), x_1 \right), \tag{21}$$

$$c^i(\mathbf{x}) = \begin{cases} r(\mathbf{x}) x_{1+i} + l^i(x_1) w(\mathbf{x}) - \theta_a(\mathbf{x})_i, & \text{for } i = 1, \ldots, N-1 \\ r(\mathbf{x}) x_{1+N} + l^N(x_1) w(\mathbf{x}), & \text{for } i = N. \end{cases} \tag{22}$$

In our setting, the first order conditions are necessary and sufficient. Therefore, any FREE constitutes a competitive equilibrium in the sense of definition 1.

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) by a deep neural network. To do so, we combine four ingredients: first, a loss function, *i.e.*, a measure of the quality of the approximation of the equilibrium functions at a given state of the economy. To evaluate our loss function, we only require states of the economy as inputs, and hence can train the neural network in an unsupervised fashion. Second, a simulation-based method for choosing the states of the economy that the algorithm uses to improve the approximation of the equilibrium functions. Third, given the loss function and a set of states, our algorithm uses mini-batch stochastic gradient descent to update the parameters of the neural network.[8] Fourth, the functional form of the chosen approximator belongs to the family of densely connected deep neural networks, which, at their core, are highly flexible function approximators.

[8] In addition to the short introduction to gradient descent given in section 4.2.2, appendix A.1 provides more information on mini-batch gradient descent.

In this section, we first detail each of the four ingredients before outlining how they are combined to form a deep equilibrium net. The combination of the four ingredients allows us to a) use a large number of states to assess the quality of our approximation,[9] 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.

4.2.1 A loss function for deep equilibrium nets

To repeat, 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, *i.e.*, 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 $\mathbf{x}$ to the approximated equilibrium functions

$$\mathcal{N}_{\boldsymbol{\rho}} : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{2(N-1)} : \tag{23}$$

$$\mathbf{x} = \begin{bmatrix} z \\ \mathbf{k} \end{bmatrix} \to \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}(\mathbf{x}) = \begin{bmatrix} \hat{\boldsymbol{\theta}}_a(\mathbf{x}) \\ \hat{\boldsymbol{\theta}}_\lambda(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \hat{\theta}_a(\mathbf{x})_1 \\ \dots \\ \hat{\theta}_a(\mathbf{x})_{N-1} \\ \hat{\theta}_\lambda(\mathbf{x})_1 \\ \dots \\ \hat{\theta}_\lambda(\mathbf{x})_{N-1} \end{bmatrix} =: \begin{bmatrix} \hat{a}^1(\mathbf{x}) \\ \dots \\ \hat{a}^{N-1}(\mathbf{x}) \\ \hat{\lambda}^1(\mathbf{x}) \\ \dots \\ \hat{\lambda}^{N-1}(\mathbf{x}) \end{bmatrix}. \tag{24}$$

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 approximately for all $i = 1, \dots, N-1$

$$u'(\hat{c}^i(\mathbf{x})) = \beta \mathrm{E}_z \left[r(\hat{\mathbf{x}}_+) u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_+)) \right] + \hat{\lambda}^i(\mathbf{x}), \tag{25}$$

$$\hat{\lambda}^i(\mathbf{x})(\hat{a}^i(\mathbf{x}) - \underline{a}) = 0, \tag{26}$$

$$(\hat{a}^i(\mathbf{x}) - \underline{a}) \geq 0, \tag{27}$$

$$\hat{\lambda}^i(\mathbf{x}) \geq 0 \tag{28}$$

holds, where

$$\hat{\mathbf{x}}_+ = \begin{bmatrix} z_+ \\ 0 \\ \hat{a}^{[1:N-1]}(\mathbf{x}) \end{bmatrix}, \tag{29}$$

and where z_+ denotes the random exogenous shock in the next period. Furthermore, $r(\cdot)$ and $w(\cdot)$ are defined according to equations (20) and (21), respectively, and

$$\hat{c}^i(\mathbf{x}) = \begin{cases} r(\mathbf{x}) x_{1+i} + l^i(x_1) w(\mathbf{x}) - \hat{a}^i(\mathbf{x}), & \text{for } i = 1, \dots, N-1 \\ r(\mathbf{x}) x_{1+N} + l^N(x_1) w(\mathbf{x}), & \text{for } i = N. \end{cases} \tag{30}$$

The sole difference between definition 2 and equations (25) – (30) is that now, the equilibrium functions $\boldsymbol{\theta}(\cdot)$ are replaced by $\mathcal{N}_{\boldsymbol{\rho}}(\cdot)$. If the neural network $\mathcal{N}_{\boldsymbol{\rho}}(\cdot)$ would encode the true equilibrium functions exactly, equations (25) – (30) would hold exactly for all states $\mathbf{x}$ of the economy. Therefore, we can define a measure of the quality of the approximation $\mathcal{N}_{\boldsymbol{\rho}}(\cdot)$ by evaluating equations (25) – (30) for one or multiple given states $\mathbf{x}$ and quantify the extent to which they are fulfilled. In the context of deep learning, such a measure is often referred to as a *loss function* or a *cost function* where lower values correspond to lower errors in the desired conditions.

[9] We use more than a billion of simulated states to train the neural network.

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 set of states at which the desired conditions are evaluated during trainingis referred to as the *training set*, which we denote by $\mathcal{D}_{\text{train}}$.

Given parameters $\boldsymbol{\rho}$ and a set of states $\mathcal{D}_{\text{train}}$, we define the loss function as:

$$\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|}\frac{1}{N-1}\sum_{\mathbf{x}_j \in \mathcal{D}_{\text{train}}}\sum_{i=1}^{N-1}\left(\left(e^i_{\text{EE}}(\mathbf{x}_j)\right)^2 + \left(e^i_{\text{KKT}}(\mathbf{x}_j)\right)^2\right), \tag{31}$$

where

$$e^i_{\text{EE}}(\mathbf{x}_j) := u'(\hat{c}^i(\mathbf{x}_j)) - \beta \text{E}_{z_j}\left[r(\hat{\mathbf{x}}_{j,+})u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+}))\right] - \hat{\lambda}^i(\mathbf{x}_j), \tag{32}$$

$$e^i_{\text{KKT}}(\mathbf{x}_j) := \hat{\lambda}^i(\mathbf{x}_j)(\hat{a}^i(\mathbf{x}_j) - \underline{a}), \tag{33}$$

are the Euler equation errors and KKT slackness errors, respectively, where

$$\hat{\mathbf{x}}_{j,+} = \begin{bmatrix} z_+ \\ 0 \\ \hat{a}^{[1:N-1]}(\mathbf{x}_j) \end{bmatrix}, \tag{34}$$

where z_+ denotes the random exogenous shock in the next period. Moreover, $r(\cdot)$, $w(\cdot)$, and $\hat{c}^i(\cdot)$ are defined according to equations (20), (21), and (30), respectively, and

$$\begin{aligned} \hat{a}^i(\mathbf{x}_j) - \underline{a} \geq 0, \text{ for } i = 1, \ldots, N-1, \\ \hat{\lambda}^i(\mathbf{x}_j) \geq 0, \text{ for } i = N, \ldots, 2(N-1). \end{aligned} \tag{35}$$

Conditions (35) must be appended to the loss function unless a functional form of the function approximator is chosen that guarantees that said conditions always hold. Following Judd (1992), we convert the Euler equation error of generation i in state $\mathbf{x}_j$—that is, $e^i_{\text{EE}}(\mathbf{x}_j)$—to the relative Euler equation error defined as

$$e^i_{\text{REE}}(\mathbf{x}_j) := \frac{u'^{-1}\left(\beta \text{E}_{z_j}\left[r(\hat{\mathbf{x}}_{j,+})u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+}))\right] + \hat{\lambda}^i(\mathbf{x}_j)\right)}{\hat{c}^i(\mathbf{x}_j)} - 1. \tag{36}$$

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. By noting that

$$e^i_{\text{EE}}(\mathbf{x}_j) = 0 \Leftrightarrow e^i_{\text{REE}}(\mathbf{x}_j) = 0, \tag{37}$$

we can redefine the loss function as:

$$\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|}\frac{1}{N-1}\sum_{\mathbf{x}_j \in \mathcal{D}_{\text{train}}}\sum_{i=1}^{N-1}\left(\left(e^i_{\text{REE}}(\mathbf{x}_j)\right)^2 + \left(e^i_{\text{KKT}}(\mathbf{x}_j)\right)^2\right). \tag{38}$$

4.2.2 Learning the network parameters

This section describes how to use the loss function, *i.e.*, the measure for the quality of the approximation of the equilibrium functions $\mathcal{N}_{\boldsymbol{\rho}}(\cdot)$, defined in equation (38), to optimize the parameters $\boldsymbol{\rho}$.

We defined the loss function 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, gradient descent, a first-order method, is typically used to optimize the parameters $\boldsymbol{\rho}$. Gradient descent updates the parameters step-wise in the direction in which the loss function decreases—that is:

$$\rho_k^{\text{new}} = \rho_k^{\text{old}} - \alpha^{\text{learn}} \frac{\partial \ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}^{\text{old}})}{\partial \rho_k^{\text{old}}} \; \forall k \in \{1, \ldots, length(\boldsymbol{\rho})\}. \tag{39}$$

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 A.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 obtain new and large training sets $\mathcal{D}_{\text{train}}$ (introduced in the next section) enables our algorithm to circumvent this bottleneck.

4.2.3 Sampling the most relevant states

As described in the previous section 4.2.2, the parameters of the neural network $\boldsymbol{\rho}$ are chosen to minimize the loss function (defined by eq. (38)). 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, it is essential that they provide a good approximation for those states, which will be visited during the simulation. Therefore, we chose to train 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.[10] Since our method approximates the equilibrium functions directly, simulating the evolution of the economy is computationally very 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 (39).[11] 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 an almost arbitrary amount of training data in this setting, over-fitting is not a primary concern. As the neural network learns better parameter values, the simulated states become better representatives of the ergodic set of states of the economy.

Note that, 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 improve the approximation quality at specific areas of the state space, or if one is interested in a specific sequence of exogenous shocks, the training set $\mathcal{D}_{\text{train}}$ can easily be chosen accordingly.

[10] 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 B.2.

[11] More precisely, we use Adam (Kingma and Ba, 2014)—the de facto standard in the field of deep learning. For the difference between gradient descent and mini-batch gradient descent, see appendix A.1.

4.2.4 Functional form of deep neural networks

We use densely connected deep neural networks as a function approximator in our applications below. A neural network is a class of machine learning models that recently have found application in a wide range of fields ranging from science and engineering to finance (see, *e.g.*, Goodfellow et al., 2016 and references therein for a thorough introduction).

The parameters of a densely connected neural network are a set of weight matrices $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_K\}$, where $\mathbf{W}_i \in \mathbb{R}^{m_i \times m_{i+1}}$, and a set of bias vectors $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_K\}$, where $\mathbf{b}_i \in \mathbb{R}^{m_{i+1}}$ and where m_i, the number of nodes in the i^{th} layer, is chosen by the modeler. In the notation introduced above, $\boldsymbol{\rho}$ is the flattened aggregation of all weight matrices in $\mathbf{W}$ and bias vectors in $\mathbf{b}$ into a single vector, allowing the indexing of the parameter vector $\boldsymbol{\rho}$. Given the weight matrices and bias vectors, a neural network maps: $\mathbf{X} \to \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{X}; \mathbf{W}, \mathbf{b}) = \sigma_K(...\sigma_2(\sigma_1(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2)...\mathbf{W}_K + \mathbf{b}_K)$, where σ_i are non-linear activation functions that are also chosen by the modeler.

Neural networks have several favorable properties: they are universal function approximators (Hornik et al., 1989), can resolve local and highly non-linear features, can handle a large amount of high-dimensional input data, and have shown potential to overcome the curse of dimensionality to some extent when solving PDEs and computing optimal stopping rules (Grohs et al., 2018; Jentzen et al., 2018; Becker et al., 2018; Sirignano and Spiliopoulos, 2018).

4.2.5 Putting it all together

In this section, we summarize how the ingredients described in the previous sections are combined to search for approximate recursive equilibria.

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 of states and improving the parameters $\boldsymbol{\rho}$ of the neural network by performing 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 $N^{\text{epochs}} = 1$. Consequently, each simulated state is only used for a single gradient descent step, which guards our algorithm against overfitting.[12] Algorithm 1 provides the pseudo code of the deep equilibrium net. For simplicity, the pseudo code implement gradient descent rather than mini-batch gradient descent.

Note that every ingredient and the interactions between the ingredients are important to our algorithm: the loss function merely requires a set of states of the economy to be evaluated. Since only a set of states is required to evaluate the loss function, we can repeatedly generate new and large training sets sampled from the approximated ergodic distribution of states of the economy at virtually zero computational cost. In that way, we can use neural networks with millions of parameters without the risk of overfitting to specific states. Allowing for a large amount of parameters on the other hand is important for obtaining enough flexibility to approximate high-dimensional functions with strong non-linearities and local features. Given the large amount of training data and the large amount of parameters, (mini-batch) gradient descent, a first order method, enables us to find good parameter values without exhausting computational resources.

[12] More precisely, we perform one mini-batch gradient descent step on each data point, see appendix A.1 for details.

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**

$\mathcal{D}^i_{\text{train}} \leftarrow \{\mathbf{x}^i_1, \mathbf{x}^i_2, \ldots, \mathbf{x}^i_T\}$ (generate new training data by simulating an episode of T periods as implied by the parameters $\boldsymbol{\rho}^i$)

$\mathbf{x}^{i+1}_0 \leftarrow \mathbf{x}^i_T$ (set new starting point)

$\epsilon_{\max} \leftarrow \max\left\{\max_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e_{\text{REE}}(\mathbf{x})|, \max_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e_{\text{KKT}}(\mathbf{x})|\right\}$ (calculate max error on new data)

$\epsilon_{\text{mean}} \leftarrow \max\left\{\frac{1}{T}\sum_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e_{\text{REE}}(\mathbf{x})|, \frac{1}{T}\sum_{\mathbf{x}\in\mathcal{D}^i_{\text{train}}} |e_{\text{KKT}}(\mathbf{x})|\right\}$ (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**

$$\rho^{i+1}_k = \rho^i_k - \alpha^{\text{learn}} \frac{\partial \ell_{\mathcal{D}^i_{\text{train}}}(\boldsymbol{\rho}^i)}{\partial \rho^i_k} \tag{40}$$

(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 $\leftarrow True, \boldsymbol{\rho}^{\text{final}} \leftarrow \boldsymbol{\rho}^i$) ;

Algorithm 1: Algorithm for training deep equilibrium nets.

5 Performance of the algorithm

It is a well-known fact that the saving behavior of economic agents differs over the life-cycle. Wong (2016), for instance, shows that the difference in saving behavior across age-groups drives differences in the consumption response to monetary policy shocks. Similarly, it is to be expected

	Discount factor β	Relative risk aversion γ	Capital share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1}=1.15\|\eta_t=1.15)$ $P(\eta_{t+1}=0.85\|\eta_t=0.85)$	Persistence depreciation $P(\delta_{t+1}=0.5\|\delta_t=0.5)$ $P(\delta_{t+1}=0.9\|\delta_t=0.9)$	Borrowing constraint $\underline{a}$
Benchmark model	0.95	2	0.3	$\{0.85, 1.15\}$	$\{0.5, 0.9\}$	0.5 0.5	0.98 0.75	0.0
Relaxed constraints	0.95	2	0.3	$\{0.85, 1.15\}$	$\{0.5, 0.9\}$	0.5 0.5	0.98 0.75	-0.5

Table 1: Parameterization of the benchmark model and the model with relaxed borrowing constraints.

that borrowing constraints impact age-groups' consumption responses differently. To investigate this heterogeneity, we study annually calibrated OLG models as outlined in section 3. These are, in addition to being interesting to study for reasons laid out, very difficult to solve computationally; the agents live for $N = 60$ periods, the endogenous state is 59 dimensional, there are aggregate shocks, and there are binding constraints. Thus, the model described in section 3 offers an excellent setting to illustrate the performance of our novel solution method for an economically, highly relevant class of models.

5.1 The setup

5.1.1 Model calibration

In this section, we briefly detail the parameterization we choose for the benchmark model and the case with relaxed borrowing constraints.

In our benchmark model, we study the case when borrowing is prohibited entirely, *i.e.*, $\underline{a} = 0$. The depreciation δ takes one of two values $\delta \in \{0.5, 0.9\}$. The transition matrix is given by

$$\Pi^{\delta} = \begin{bmatrix} 0.98 & 0.25 \\ 0.02 & 0.75 \end{bmatrix}. \tag{41}$$

In our parameterization, the high depreciation state has an average duration of 4 years and occurs approximately 7.4 years per century. The TFP takes on of two values $\eta \in \{0.85, 1.15\}$ and evolves i.i.d. with a transition matrix

$$\Pi^{\eta} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}. \tag{42}$$

Depreciation and TFP evolve independently from each other. There are in total four possible combinations of depreciation shock and TFP shock. They are labeled by $z \in \mathcal{Z} = \{1, 2, 3, 4\}$ and correspond to values $\eta = [0.85, 1.15, 0.85, 1.15]^T$ and $\delta = [0.5, 0.5, 0.9, 0.9]^T$. The overall transition matrix is given by

$$\Pi = \Pi^{\delta} \otimes \Pi^{\eta} = \begin{bmatrix} 0.49 & 0.49 & 0.125 & 0.125 \\ 0.49 & 0.49 & 0.125 & 0.125 \\ 0.01 & 0.01 & 0.375 & 0.375 \\ 0.01 & 0.01 & 0.375 & 0.375 \end{bmatrix}, \tag{43}$$

where $\Pi_{j,i}$ denotes the probability of a transition from the exogenous shock $z = i$ to the exogenous shock $z = j$. The labor endowment of the life-cycle is shown in figure 2. The figure displays that agents are "born" with age 21 and a labor endowment of 0.6. Thereafter, the labor endowment increases linearly to its maximum value of 1.4 by age 41 and stays constant at that level until age 61. Afterwards, the labor endowment decays linearly down to 0.7 by age 71 and stays there for the remaining lifetime. Capital share of production is set to $\alpha = 0.3$. We study

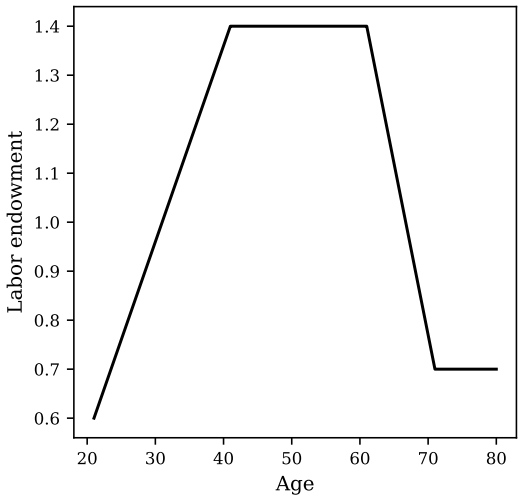


Figure 2: Labor endowment over the life-cycle.

an economy with $N = 60$ cohorts. The patience is given by $\beta = 0.95^{\frac{60}{N}} = 0.95$, and the agents' utility from consumption exhibits constant relative risk aversion $\gamma = 2$. Further details on the parameterization are summarized in table 1.

5.1.2 Architecture of the neural network

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. For example, in addition to the distribution of capital and the exogenous shock, we augment the state of the economy with the distribution of labor endowments, labor income, financial income, and total income as well as aggregate variables even though the latter variables are uniquely pinned down by the former two.[13]

As a result, the input layer consists of $12 + 5 \cdot N$ input nodes, which for $N = 60$ results in 312 input nodes. After the input layer, the neural network features two hidden layers with 1,000 relu-activated hidden nodes each. The output layer consists of $2 \cdot (N - 1)$ nodes, activated with softplus functions to ensure that the non-negativity constraints on the savings and the KKT multiplier (both given by eq. (35)) are fullfilled. A schematic illustration of the neural network architecture, for clarity displayed without providing redundant information, is given in figure 3.[14]

5.1.3 Hyper-parameters

In analogy to the previous section 5.1.2, several hyperparameters were set based on numerical experimentation–a situation that one typically faces when working with neural networks. They are listed in table 2. Note that we did not find a significant variation in performance regarding the number of periods per episode T or the number of epochs per episode N^{epochs}; the latter of which we always kept 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.[15]

[13] See appendix B.1 for more details.

[14] Note that the network architecture presented here showed the best performance. As typical when working neural networks, the architecture presented here is not the result of a theoretically motivated performance optimization, but based on numerical experimentation.

[15] See appendix A.1 for a brief introduction to mini-batch stochastic gradient descent.

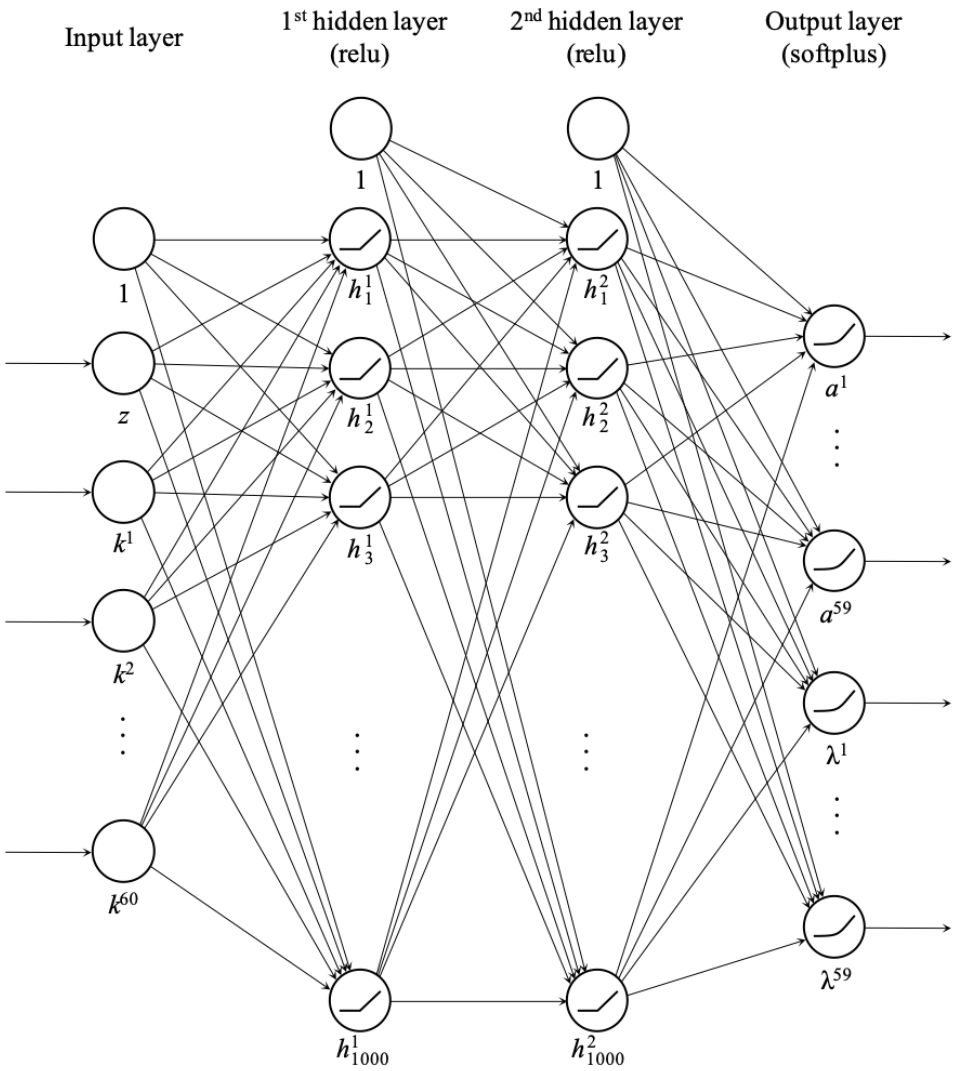


Figure 3: Schematic illustration of the neural network architecture for the deep equilibrium net. For simplicity, it is illustrated for the case where no redundant information is passed to the neural network. See appendix B for explanations on providing additional inputs to the neural network.

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	$10,000$	10	1000	1000 1000	relu relu

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

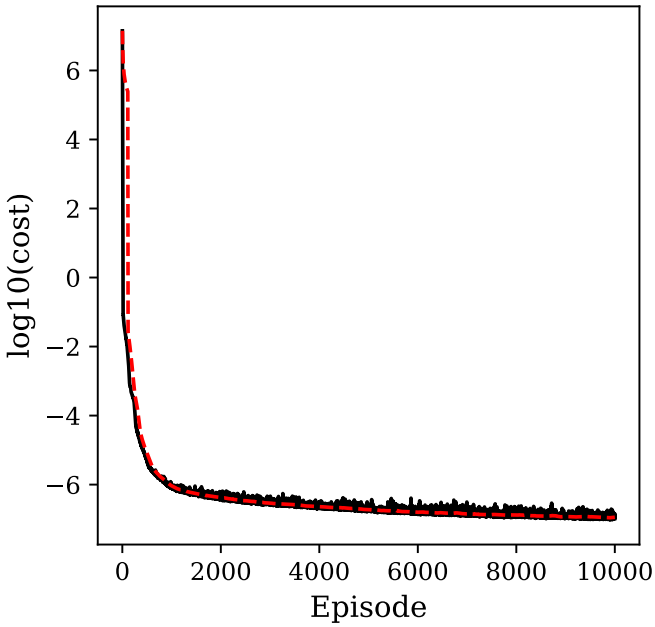


Figure 4: Evolution of the cost function during 10,000 episodes of training, 10 epochs per episode. The corresponding neural network parameters are the described in section 5.1

If the goal were to optimize the runtime of our algorithm, it might be helpful to start with a larger learning rate and decrease it over time. The number of training epochs per episode trades off the computing time spent on updating the network parameters and simulating new, more relevant, states. With the hyper-parameters specified in table 2 and not optimizing for runtime, the algorithm (including simulation, evaluation of the errors, and subsequent learning) takes approximately 10.1 seconds per episode when running on a single node 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. Instead of simulating one path with 10,000 periods, we can also simulate 1,000 different paths with ten periods each to speed up the simulation, reducing the runtime to 2.1 seconds per episode.

5.2 Training progress of the deep equilibrium net

Next, we illustrate the performance of the proposed algorithm for our benchmark model in which borrowing is not allowed (*i.e.*, $\underline{a} = 0$). Figure 4 shows the evolution of the cost function during the first 10,000 episodes of training: the cost function decreases quickly during training, and converges at approximately $10^{-6.9}$. Figure 5 shows the distribution of capital, the KKT multipliers, the consumption, and the relative Euler equation errors on a simulated path after 100, 500, and 10,000 episodes of training. It can be seen that, as the training process advances, the relative Euler equation errors (fourth row) decrease while the consumption (third row) becomes smoother over the life-cycle.

Table 3 reports the mean and maximum relative Euler equation error (in %) and error in the KKT conditions ($\times 10^{-2}$) as well as several percentiles on a simulates path of 10^4 periods, after training.[16] The mean relative error in the Euler equations as well as the KKT 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. The 99.9th percentile of both errors is below 1%. Figure 6 shows

[16]Following the literature, we simulated a total of 12,000 periods and discarded the first 2,000 "burn-in" periods.

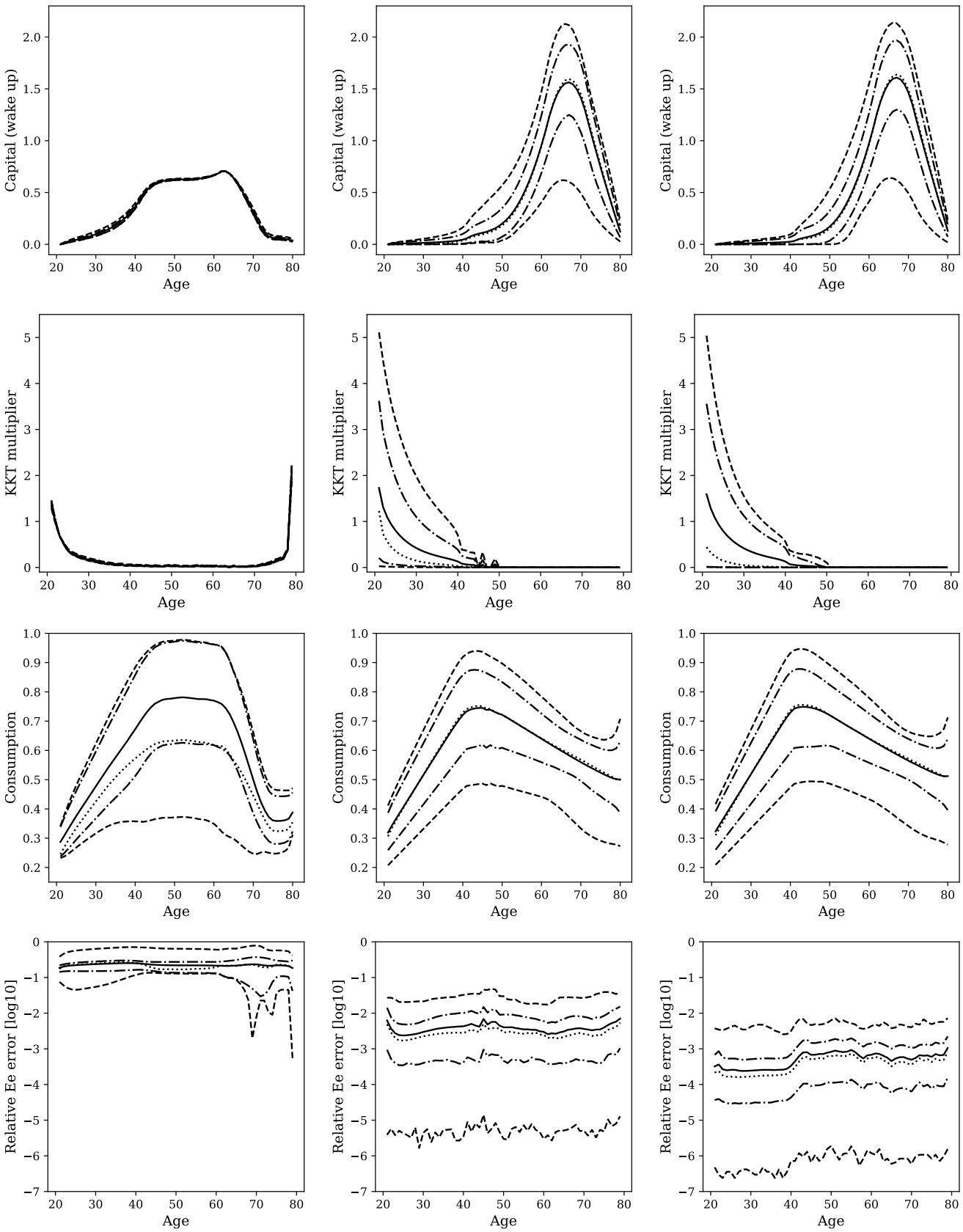


Figure 5: Distribution of capital (first row), KKT multipliers (second row), consumption (third row), and relative Euler equation errors (fourth row) on a simulated path after learning for 100 (left column), 500 (middle column), and 10,000 (right column) episodes. The solid line shows the mean over 10,000 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.

	mean	max	0.1	10	50	90	99.9
Rel Ee [%]	0.06	1.31	0.00	0.01	0.04	0.13	0.54
KKT [$\times 10^{-2}$]	0.01	0.25	0.00	0.00	0.00	0.02	0.15

Table 3: Summary statistics of the relative Euler equation errors (Rel Ee) and the errors in the KKT conditions (KKT) on 10,000 simulated periods. The summary statistics computed are, the mean errors, the max errors, and the 0.1st, 10th, 50th, 90th, and 99.9th percentiles.

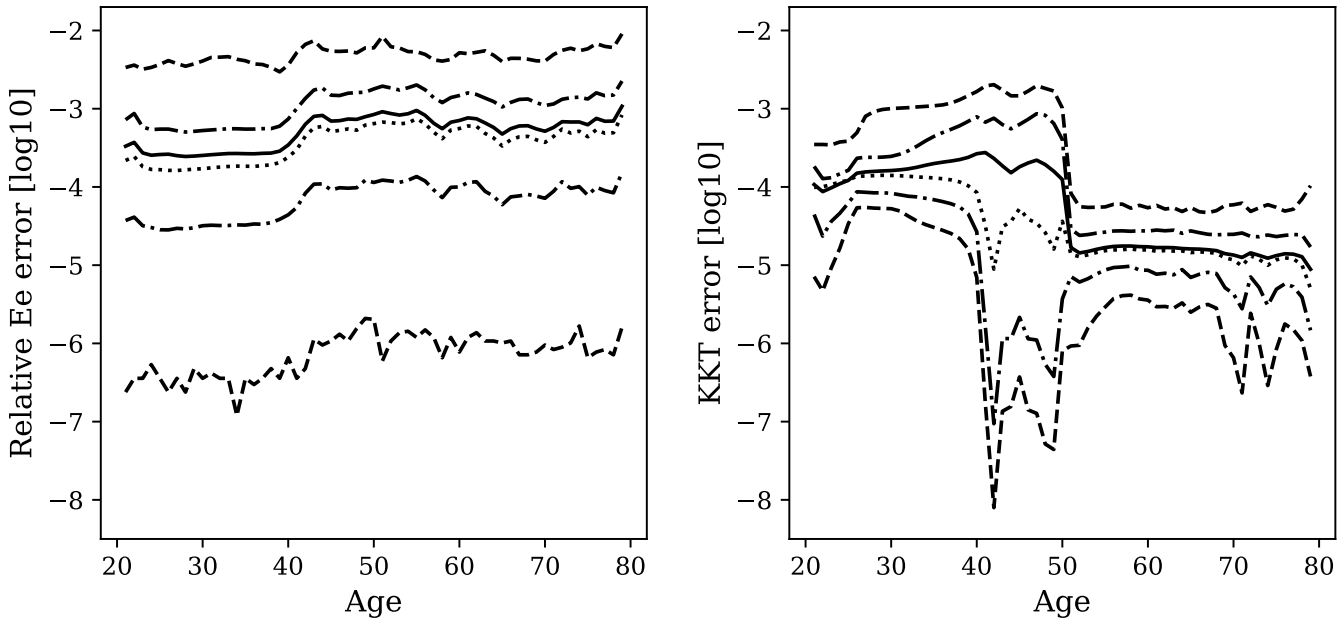


Figure 6: Distribution of the relative Euler equation error (left) and error in the KKT conditions (right). The solid line shows the mean over 10,000 simulated periods, the dotted line shows the median, the dash-dotted lines show the 10th and 90th percentile, the dashed lines show the 0.1th and 99.9th percentile.

the errors for different age-groups separately.

6 Comparison between different levels of the borrowing constraint

One key question we want to study now is how the effect of borrowing constraints on the consumption varies across age-groups in the OLG model. As demonstrated in section 5, the algorithm we propose is, in contrast to prior work (*cf.* section 2), able to compute an approximation to the functional rational expectations equilibrium of the economy as described in section 3. In this section, we use the ability to solve such a model to study how the cross-sectional consumption response to aggregate shocks depends on the level of the exogenous borrowing constraint. To do so, we compare the benchmark model in which borrowing is prohibited entirely, to the case where borrowing is allowed up to an exogenous level $a_t^s \geq \underline{a} = -0.5$. The mean consumption per period and per age-group in the model without borrowing is approximately 0.60. Thus, a lower bound on the capital holding of -0.5 corresponds to approximately -84% of average per period consumption.

Table 4 reports the mean and maximum relative Euler equation error (in %) and error in the KKT conditions ($\times 10^{-2}$), as well as several percentiles on a, simulates path of 10^4 periods, after

	mean	max	0.1	10	50	90	99.9
Rel Ee [%]	0.04	3.55	0.00	0.00	0.03	0.09	0.47
KKT [$\times 10^{-2}$]	0.01	0.42	0.00	0.00	0.00	0.02	0.14

Table 4: Summary statistics of the relative Euler equation errors (Rel Ee) and the errors in the KKT conditions (KKT) on 10,000 simulated periods for the model in which the borrowing constraint is relaxed to $\underline{a} = -0.5$. The summary statistics computed are the mean errors, the max errors, and the 0.1st, 10th, 50th, 90th, and 99.9th percentiles.

training.[17] The mean relative error in the Euler equation, in the case when some borrowing is allowed, also remains below 0.1%. Figure 7 compares the mean financial wealth, total income, and consumption over the life-cycle conditional on the exogenous shock. Financial wealth refers to the value of the capital saved in the last period after returns realized. The total income refers to the sum of the financial wealth and the labor-income in the corresponding period. In the setting with relaxed borrowing constraints, the young agents take up debt that they pay back as their labor endowment increases in their later years. Whether the borrowing constraint binds depends on the sequence of exogenous shocks experienced by an agent. Allowing agents to take up debt affects the economy's response to shocks since young agents benefit from a low interest rate on capital. In such a model, any shock to the return on capital, which results in a shift of wealth from lenders to borrowers, or vice versa, implies a shift of wealth between age-groups.

To see how this influences aggregate quantities, we investigate the impulse response of the aggregate consumption in response to each of the four shocks. To obtain the impulse response, we simulate 50 periods, once with a specific shock happening in period 25, and once without. We repeat this 200,000 times and compute the mean of the aggregate consumption in each period for both cases. Afterward, we compare the mean values of the aggregate consumption, conditional on a specific shock occurring in period 25, to the unconditional mean values. The results of this exercise are displayed in figure 8. We find that the reaction of the aggregate consumption is always stronger in the case without borrowing. This is explained by the fact that in the case without borrowing, agents are more limited in their ability to smooth their consumption. On impact, the depreciation shock affects only the return, while the TFP shock affects wages as well. However, the persistence of the depreciation shock affects the wages in the following periods through a reduction of the aggregate capital in the economy. The combination of these two effects explains the impulse response for shock 4 (fig. 8d), where both depreciation and TFP are high. On impact, wages increase due to the TFP shock. Young agents consume more and drive aggregate consumption above the unconditional average. Additionally, in the case in which some borrowing is allowed, the indebted young agents have to pay a much lower interest rate. Since the depreciation is, with a very high probability (75% of the time), followed by another high depreciation shock, aggregate savings decrease. As a result, aggregate capital in the next period and, consequently, wages in the next period, decrease as well.

Using the same simulations as before, we next study the consumption response at the time of the exogenous shock across age-groups. Figure 9 shows the cross-sectional reaction of consumption when exposed to each of the four shocks. When hit by either of the low depreciation shocks (shocks 1 and 2 presented in figs. 9a and 9b, respectively), the consumption of the young agents reacts much stronger in the case without borrowing. Even though young agents have a small amount of precautionary savings accumulated during times of high TFP, from which they can consume when the TFP is low, the borrowing constraint still often binds when the TFP is low. As labor income comprises the majority of a young agent's income, the propensity to consume out of the (higher or lower) transitory labor income is large. Since, in the early years, the labor income increases with age, young agents do not have an incentive to accumulate larger precautionary savings which would prevent the constraint from binding. In the case

[17] We simulated a total of 12,000 periods and discarded the first 2,000 "burn-in" periods.

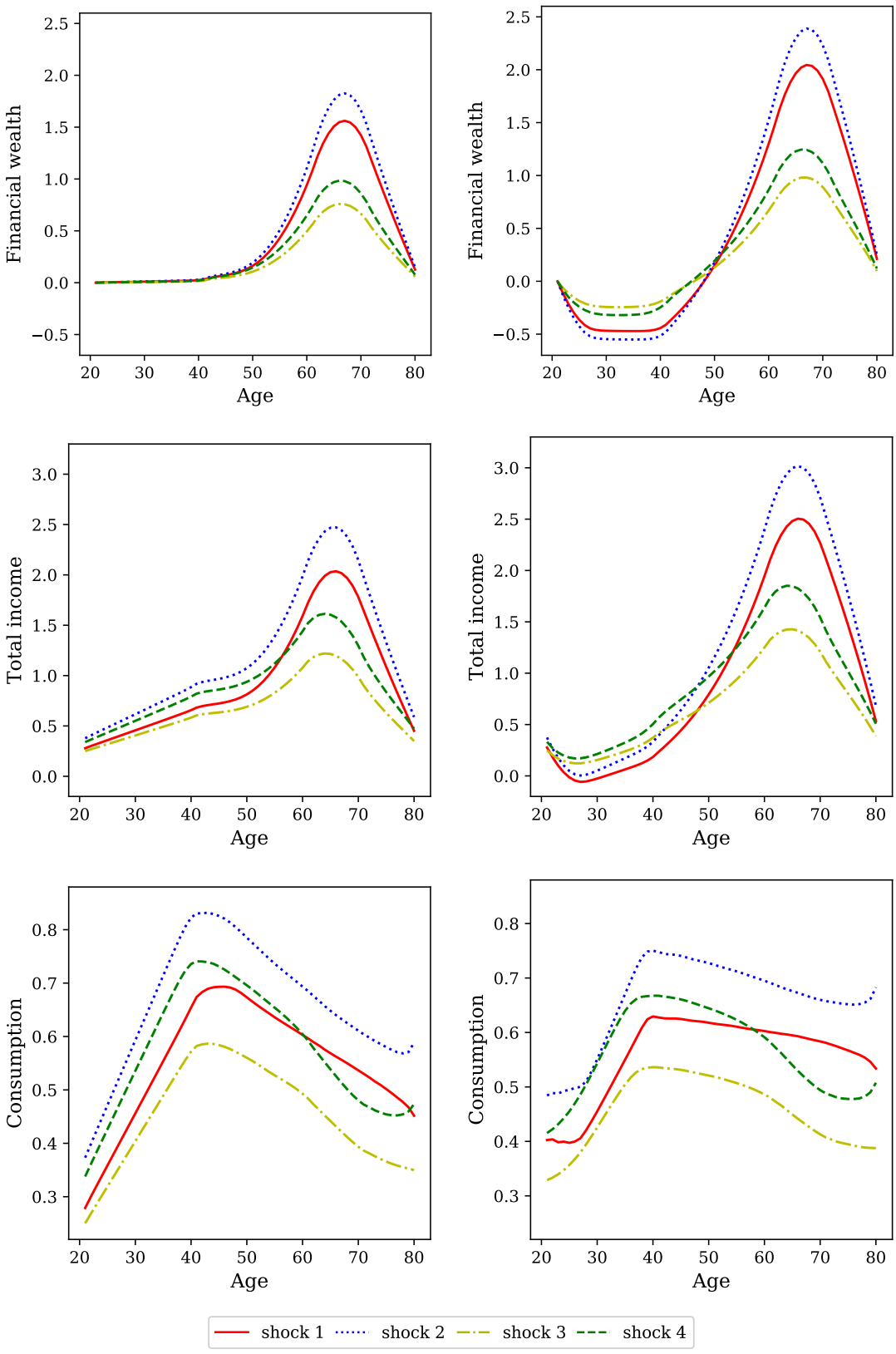
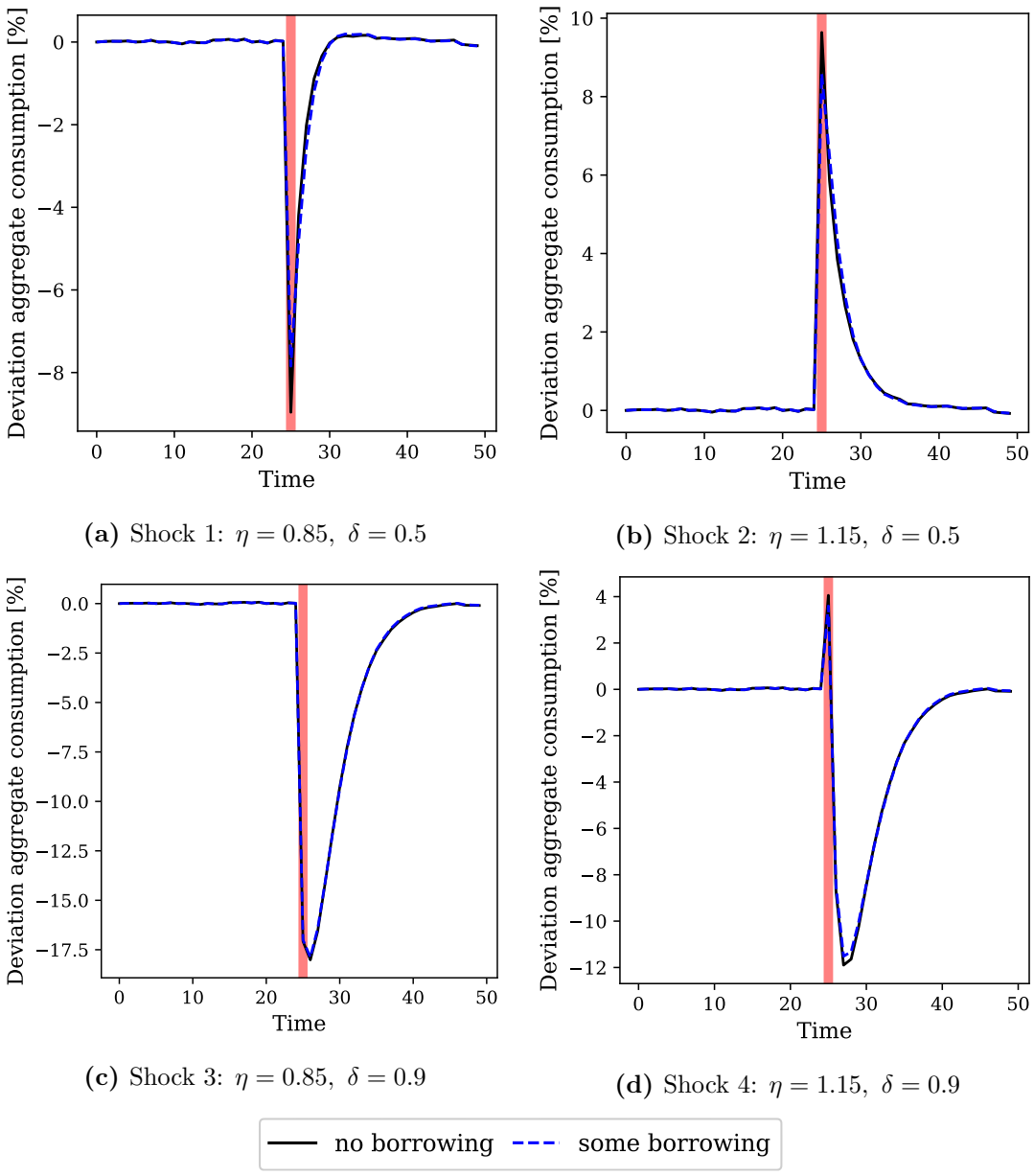


Figure 7: The first, second, and third row displays the distribution of mean financial wealth, mean total income, and mean consumption on a simulated path of 10'000 periods, respectively. The left column presents the benchmark case in which the borrowing constraint is set to $\underline{a} = 0$ and the right column presents the case in which the borrowing constraint is relaxed to $\underline{a} = -0.5$. The TFP and deprecation $(\eta,\ \delta)$ of shock 1, shock 2, shock 3, and shock 4 are $(0.85,\ 0.5)$, $(1.15,\ 0.5)$, $(0.85,\ 0.9)$, and $(1.15,\ 0.9)$, respectively.



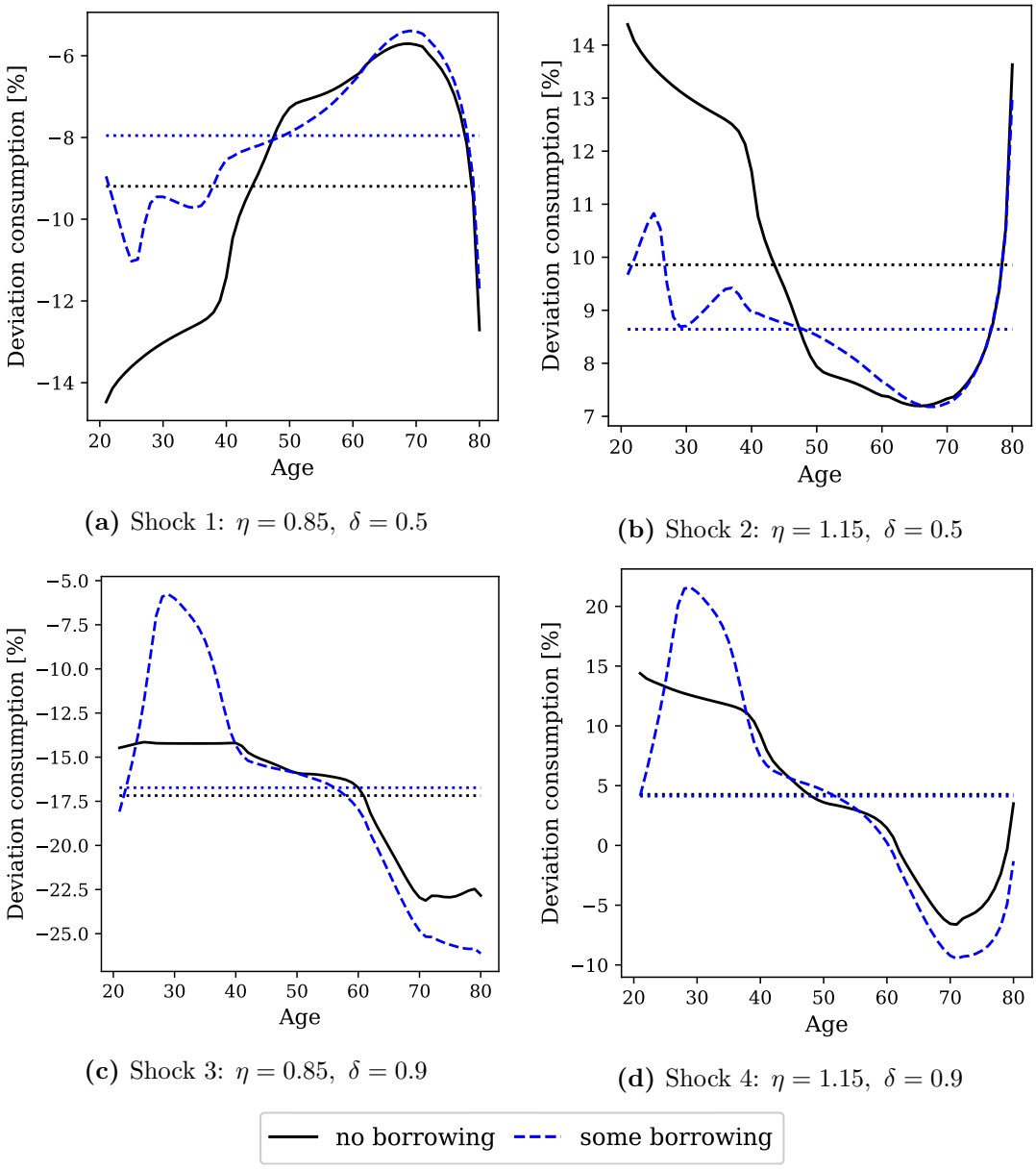
(a) Shock 1: $\eta = 0.85,\ \delta = 0.5$

(b) Shock 2: $\eta = 1.15,\ \delta = 0.5$

(c) Shock 3: $\eta = 0.85,\ \delta = 0.9$

(d) Shock 4: $\eta = 1.15,\ \delta = 0.9$

Figure 8: Mean percent deviation of the aggregate consumption from its unconditional average when each of the exogenous shocks occurs in period 25. The mean impulse response was computed over 200,000 simulations. In the case without borrowing (solid black line), the constraint is set to $\underline{a} = 0$. In the case with some borrowing (dashed blue line), it is relaxed to $\underline{a} = -0.5$. On impact of the respective shock, aggregate consumption in the case without borrowing and in the case with some borrowing change as follows: (a) decreases by 9.0% and 7.8%; (b) increases by 9.6% and 8.6%; (c) decreases by 17.1 % and 17.0 %; and (d) increases by 4.1 % and 3.6 %, respectively.



(a) Shock 1: $\eta = 0.85,\ \delta = 0.5$

(b) Shock 2: $\eta = 1.15,\ \delta = 0.5$

(c) Shock 3: $\eta = 0.85,\ \delta = 0.9$

(d) Shock 4: $\eta = 1.15,\ \delta = 0.9$

Figure 9: Mean percent deviation of consumption by age-group from its unconditional average when each of the exogenous shocks occurs. The mean is taken over 200,000 simulations. The solid black line shows the case in which the borrowing constraint is set to $\underline{a} = 0$. The dashed blue line shows the case in which the borrowing constraints is relaxed to $\underline{a} = -0.5$.

where borrowing is allowed, the consumption of agents around the age of 25 reacts stronger than the consumption of agents aged 21. This comes from the fact that, in the model where some borrowing is allowed, 21 year old agents are unconstrained while the constraint only starts to be binding around age 25. Shocks with persistently high depreciation (shocks 3 and 4 presented in figs. 9c and 9d, respectively) are especially bad for the older agents who hold their wealth in capital. Since these agents hold more capital in the case with borrowing, their reaction is stronger. In contrast, high depreciation is comparatively good news for young, indebted agents (*i.e.*, in the case with borrowing), especially in combination with high TFP. Relative to their less indebted counterparts, heavily indebted agents reduce their consumption by a lesser amount for shock 3 and increase it by a larger amount for shock 4. The same is true in contrast to the model in which borrowing is not allowed.

To summarize, we find substantial heterogeneity in the effect of looser borrowing constraints across age-groups; an effect that was, to the best of our knowledge, not possible to study quantitatively with that level of detail prior to this work.

7 Illiquid capital and a liquid bond

In this section, we investigate how the cross-sectional consumption is affected if agents can choose between saving in an illiquid or a liquid asset. As highlighted in Kaplan et al. (2018), such a distinction is essential for studying cross-sectional consumption, which depends on the liquidity of the assets held by different agents. To do so, we enrich the benchmark model from above in two ways: first, we impose convex adjustment costs on capital holdings, which render capital an illiquid asset; second, we allow agents to trade a liquid one-period bond subject to collateral constraints.

7.1 The economic model

As in the benchmark model outlined in section 5.1.1, the agents can save in risky capital. The model presented in this section differs in that the investment in capital is subject to adjustment costs, rendering capital an illiquid asset. Additionally, agents can trade a liquid, collateralized, one-period bond in zero net supply. Capital cannot be sold short. To short-sell the bond, agents must post capital as collateral. As before, $a_t^s = k_{t+1}^{s+1}$ denotes the investment in capital by agent s in period t. Similarly, let $d_t^s = b_{t+1}^{s+1}$ denote the investment in the one-period bond by agent s at time t. The collateral requirement is based on Kubler and Schmedders (2003): in order to short sell d_t units of the bond in period t, $\kappa \cdot d_t$ units of risky capital have to be pledged as collateral. The parameter κ hence governs the collateral requirement for short-selling the bond. If the value of the pledged collateral drops below the promised payout of the bond, agents default without further punishment. In our model, we chose κ high enough so that default never occurs in equilibrium, and the bond is indeed risk-free. Formally, the short selling constraint for an agent of age s is given by $\kappa \cdot d_t^s + a_t^s \geq 0$, where κ is an exogenous constant.

Capital is modeled to be an illiquid asset: changing the level of the capital holding is subject to convex adjustment costs $f(k_t^s, a_t^s) = \zeta \left(a_t^s - k_t^s\right)^2$, where ζ is an exogenous constant that governs the illiquidity of capital. The bond is modeled to be a liquid asset, *i.e.*, there are no adjustment costs. Because of the adjustment costs of capital, the endogenous state cannot be reduced to the exogenous shock and the distribution of financial wealth alone. This is because of the decomposition of the agents' financial wealth into bonds and capital matters. Hence, when modeling N age-groups, the endogenous state, composed of the capital holding and the bond holding, is approximately $2N$-dimensional. Note that the exact dimension of the endogenous state is $2N - 3$. The origin of the three redundant variables is (1) that newly born agents have

zero capital, (2) that newly born agents hold no bonds, and (3) because of bond market clearing. However, we retain these variables in the state in order to improve readability.

Apart from the described modifications, the model is the same as described in section 3. That is, we adopt the structure of the uncertainty (section 3.1) and the model of the firms (section 3.3) as is. The model of the households (section 3.2) and markets (section 3.4) need to updated to account for the existence of the one-period bond and the adjustment costs on capital. For clarity, we proceed by explicitly defining the modified functional rational expectations for this economy as well as the projection algorithm, before reporting the performance of the proposed algorithm and economic results.

7.2 Approximating the equilibrium with deep neural networks

We use the same algorithm described previously in section 4 to approximate the recursive equilibrium in this economy. Although the general idea of the algorithm remains the same, the new economics, which enters the algorithm through the equilibrium conditions in the loss function, requires some adjustments to be made. The structure of this section follows the structure of section 4 closely, but focuses on how the loss function is adjusted to encode this model.

7.2.1 Functional rational expectations equilibrium

We define a FREE for the enriched economy:[18]

Definition 3 (FREE with illiquid capital and a liquid one-period bond) *A FREE consists of equilibrium functions* $\boldsymbol{\theta} = [\boldsymbol{\theta}_a^T,\ \boldsymbol{\theta}_\lambda^T,\ \boldsymbol{\theta}_d^T,\ \boldsymbol{\theta}_\mu^T,\ \theta_p]^T : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{4(N-1)+1}$, *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, and* $\boldsymbol{\theta}_\mu : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{N-1}$ *denotes the KKT-multiplier functions for the collateral constraint, 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* $i = 1, \ldots, N-1$:[19]

$$\begin{aligned}
(1 + 2\zeta\left(\theta_a(\mathbf{x})_i - x_{1+i}\right))\, u'(c^i(\mathbf{x})) &= \beta E_z\left[\left(r(\mathbf{x}_+) + 2\zeta(\theta_a(\mathbf{x}_+)_{i+1} - x^+_{2+i})\right) u'(c^{i+1}(\mathbf{x}_+))\right] \\
&\quad + \theta_\lambda(\mathbf{x})_i + \theta_\mu(\mathbf{x})_i, && (44)\\
\theta_\lambda(\mathbf{x})_i \theta_a(\mathbf{x})_i &= 0, && (45)\\
\theta_a(\mathbf{x})_i &\geq 0, && (46)\\
\theta_\lambda(\mathbf{x})_i &\geq 0, && (47)
\end{aligned}$$

as well as

$$\begin{aligned}
\theta_p(\mathbf{x}) u'(c^i(\mathbf{x})) &= \beta E_z\left[\left(\min\{\kappa r(\mathbf{x}_+), 1\}\right) u'(c^{i+1}(\mathbf{x}_+))\right] + \kappa\theta_\mu(\mathbf{x})_i, && (48)\\
\theta_\mu(\mathbf{x})_i(\theta_a(\mathbf{x})_i + \kappa\theta_d(\mathbf{x})_i) &= 0, && (49)\\
\theta_a(\mathbf{x})_i + \kappa\theta_d(\mathbf{x})_i &\geq 0, && (50)\\
\theta_\mu(\mathbf{x})_i &\geq 0 && (51)
\end{aligned}$$

[18] We assume that in the last year of their life, the agents consume everything and do not invest. Due to the adjustment costs, this is an assumption since it could potentially be suboptimal to do so. For the parameters we use, this assumption is innocuous.

[19] With a slight abuse of notation, we set $\theta_a(\mathbf{x}_+)_N = \theta_d(\mathbf{x}_+)_N = 0$. In principle, $\theta_a(\mathbf{x}_+)$ 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.

and

$$\sum_{i=1}^{N-1} \theta_d(\mathbf{x})_i = 0, \tag{52}$$

where

$$\mathbf{x}_+ = [z_+,\ 0,\ \boldsymbol{\theta}_a(\mathbf{x})^T,\ 0,\ \boldsymbol{\theta}_d(\mathbf{x})^T]^T, \tag{53}$$

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

$$r(\mathbf{x}) = f_K\left(\sum_{i=1}^{N} x_{1+i}, \sum_{i=1}^{N} l^i(x_1), x_1\right), \tag{54}$$

$$w(\mathbf{x}) = f_L\left(\sum_{i=1}^{N} x_{1+i}, \sum_{i=1}^{N} l^i(x_1), x_1\right), \tag{55}$$

$$c^i(\mathbf{x}) = \begin{cases} r(\mathbf{x})x_{1+i} + \min\{\kappa r(\mathbf{x}), 1\}x_{1+i+N} + l^i(x_1)w(\mathbf{x}) - \theta_a(\mathbf{x})_i - \theta_p(\mathbf{x})\theta_d(\mathbf{x})_i \\ -\zeta\left(x_{1+i} - \theta_a(\mathbf{x})_i\right)^2,\ \textit{for } i = 1, \ldots, N-1 \\ r(\mathbf{x})x_{1+N} + \min\{\kappa r(\mathbf{x}), 1\}x_{1+2N} + l^N(x_1)w(\mathbf{x}) - \zeta\left(x_{1+N}\right)^2,\ \textit{for } i = N. \end{cases} \tag{56}$$

Computing an approximation to the FREE in this model here means finding an approximation for the $4 \cdot (N-1) + 1$ equilibrium functions such that the above equations are approximately fulfilled for all exogenous shocks and all values the $(2N-3)$-dimensional endogenous state takes.

7.2.2 Algorithm to train deep equilibrium nets

In this section, we describe how to apply the deep equilibrium net framework to approximate the FREE from definition 3. The procedure is analogous to the procedure for our benchmark model (see section 4.2). We repeat it here for clarity and convenience, focusing on the adjustments that stem from the new model specification.

A loss function for deep equilibrium nets As described in section 4.2.1, we want to construct a measure of the quality of our approximation at a given state of the economy. Since the equilibrium functions we wish to approximate are characterized by equations (44) – (56), we can construct such a measure using the extent to which the approximation fulfills these equations.

Let $\boldsymbol{\rho}$ denote the tunable parameters of a neural network:

$$\mathcal{N}_{\boldsymbol{\rho}} : \mathcal{Z} \times \mathbb{R}^{2N} \to \mathbb{R}^{4(N-1)+1} : \mathbf{x} \to \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}). \tag{57}$$

The goal is to numerically determine the parameters $\boldsymbol{\rho}$ such that the neural network approximates the equilibrium functions $\boldsymbol{\theta}$ characterized by conditions (44) – (56): $\mathcal{N}_{\boldsymbol{\rho}}(x) = \hat{\boldsymbol{\theta}}(\mathbf{x}) = [\hat{\boldsymbol{\theta}}_a(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\lambda(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_d(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\mu(\mathbf{x})^T,\ \hat{\theta}_p(\mathbf{x})]^T =: [\hat{\boldsymbol{a}}(\mathbf{x})^T,\ \hat{\boldsymbol{\lambda}}(\mathbf{x})^T,\ \hat{\mathbf{d}}(\mathbf{x})^T,\ \hat{\boldsymbol{\mu}}(\mathbf{x})^T,\ \hat{p}(\mathbf{x})]^T \approx \boldsymbol{\theta}(\mathbf{x})$. To this end, we construct our loss function by summing up the mean squared error of all equilibrium conditions for a set of states $\mathcal{D}_{\text{train}}$. For readability, we first define the loss function for each equilibrium condition separately. The loss function resulting from the Euler equation for capital investment of agent i, given state $\mathbf{x}_j$, and the corresponding KKT conditions is defined

by[20]

$$\begin{aligned}\ell^{k,i}_{\mathbf{x}_j}(\boldsymbol{\rho}) := \big((1+2\zeta(\hat{a}^i(\mathbf{x}_j) - x_{j,1+i}))u'(\hat{c}^i(\mathbf{x}_j))\\ -\beta \mathrm{E}_{z_j}\left[(r(\hat{\mathbf{x}}_{j,+}) + 2\zeta(\hat{a}^{i+1}(\mathbf{x}_{j,+}) - \hat{x}_{j,+,2+i}))u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+})\right]\\ -\hat{\lambda}^i(\mathbf{x}_j) - \hat{\mu}^i(\mathbf{x}_j)\Big)^2\\ + \left(\max\{0, -\hat{a}^i(\mathbf{x}_j)\}\right)^2\\ + \left(\max\{0, -\hat{\lambda}^i(\mathbf{x}_j)\}\right)^2\\ + \left(\hat{\lambda}^i(\mathbf{x}_j)\hat{a}^i(\mathbf{x}_j)\right)^2,\end{aligned} \tag{58}$$

whereas the loss function resulting from the Euler equation for bond investment of agent i, given state $\mathbf{x}_j$, and the corresponding KKT conditions is given by[21]

$$\begin{aligned}\ell^{b,i}_{\mathbf{x}_j}(\boldsymbol{\rho}) := \big(\hat{p}(\mathbf{x}_j)u'(\hat{c}^i(\mathbf{x}_j))\\ -\beta \mathrm{E}_{z_j}\left[\min\{\kappa r(\hat{\mathbf{x}}_{j,+}), 1\}u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+})\right] - \kappa\hat{\mu}^i(\mathbf{x}_j)\big)^2\\ + \left(\max\{0, -(\hat{a}^i(\mathbf{x}_j) + \kappa\hat{d}^i(\mathbf{x}_j))\}\right)^2\\ + \left(\max\{0, -\hat{\mu}^i(\mathbf{x}_j)\}\right)^2\\ + \left(\hat{\mu}^i(\mathbf{x}_j)(\hat{a}^i(\mathbf{x}_j) + \kappa\hat{d}^i(\mathbf{x}_j))\right)^2,\end{aligned} \tag{59}$$

and the loss function for the market clearing condition, given state $\mathbf{x}_j$ is given by

$$\ell^{\mathrm{mc}}_{\mathbf{x}_j}(\boldsymbol{\rho}) := \left(\sum_{i=1}^{N-1} \hat{d}^i(\mathbf{x}_j)\right)^2, \tag{60}$$

where

$$\hat{\mathbf{x}}_+ = \left[z_+,\ 0,\ \hat{\boldsymbol{a}}(\mathbf{x})^T,\ 0,\ \hat{\boldsymbol{d}}(\mathbf{x})^T\right]^T, \tag{61}$$

and where z_+ denotes the random exogenous shock in the next period. Furthermore, $r(\cdot)$ and $w(\cdot)$ are defined according to equations (54) and (54), respectively, and

$$\hat{c}^i(\mathbf{x}) = \begin{cases} r(\mathbf{x})x_{1+i} + \min\{\kappa r(\mathbf{x}), 1\}x_{1+i+N} + l^i(x_1)w(\mathbf{x}) - \hat{a}^i(\mathbf{x}) - \hat{p}(\mathbf{x})\hat{d}^i(\mathbf{x}) \\ -\zeta\left(x_{1+i} - \hat{a}^i(\mathbf{x})\right)^2, \text{ for } i = 1, \dots, N-1 \\ r(\mathbf{x})x_{1+N} + \min\{\kappa r(\mathbf{x}), 1\}x_{1+2N} + l^N(x_1)w(\mathbf{x}) - \zeta\left(x_{1+N}\right)^2, \text{for } i = N. \end{cases} \tag{62}$$

The loss function encoding all equilibrium conditions evaluated on a set of states $\mathcal{D}_{\text{train}}$ is defined as[22]

$$\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{\mathbf{x}_j \in \mathcal{D}_{\text{train}}} \left(\left(\frac{1}{N-1}\sum_{i=1}^{N-1} \ell^{k,i}_{\mathbf{x}_j}(\boldsymbol{\rho}) + \ell^{b,i}_{\mathbf{x}_j}(\boldsymbol{\rho})\right) + \ell^{\mathrm{mc}}_{\mathbf{x}_j}(\boldsymbol{\rho})\right). \tag{63}$$

[20] The non-negativity conditions (the third and fourth term in eq. (58)) can be omitted if the architecture of the neural network ensures that $\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x})_{[1:2(N-1)]} \geq 0$. We accomplished this, for example, by applying a softplus activation to the output.

[21] Also in the model presented in this section, we found it helpful to omit the non-negativity conditions in the loss function (the second and third terms in eq. (59)) and instead force compliance through softplus activations in the output layer. To do so, we approximate $\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) = [\hat{\boldsymbol{\theta}}_a(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\lambda(\mathbf{x})^T,\ \kappa\hat{\boldsymbol{\theta}}_d(\mathbf{x})^T + \hat{\boldsymbol{\theta}}_a(\mathbf{x})^T,\ \hat{\boldsymbol{\theta}}_\mu(\mathbf{x})^T,\ \hat{\theta}_p(\mathbf{x})]^T$ in order to allow the bond policy to be negative. The approximation of the bond policy is still given by the neural network via $\hat{\boldsymbol{\theta}}_d(\mathbf{x}) = \left(\left(\kappa\hat{\boldsymbol{\theta}}_d(\mathbf{x}) + \hat{\boldsymbol{\theta}}_a(\mathbf{x})\right) - \hat{\boldsymbol{\theta}}_a(\mathbf{x})\right)\frac{1}{\kappa}$.

[22] In analogy to equation (36), we again rearrange the Euler equation error in such a way that we obtain the relative Euler equation.

Discount factor β	Relative risk aversion γ	Capital share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1}=1.15\|\eta_t=1.15)$ $P(\eta_{t+1}=0.85\|\eta_t=0.85)$	Persistence depreciation $P(\delta_{t+1}=0.5\|\delta_t=0.5)$ $P(\delta_{t+1}=0.9\|\delta_t=0.9)$	Borrowing constraint $\underline{a}$	Adjustment cost capital ζ	Collateral requirement bond κ
0.95	2	0.3	$\{0.85, 1.15\}$	$\{0.5, 0.9\}$	0.5 0.5	0.98 0.75	0.0	0.5	3.0

Table 5: Chosen parameters for the model with illiquid capital and a liquid bond.

Since the economic model is entering our algorithm through the definition of the loss function, the remaining ingredients to the computational method remain unchanged. Given the loss function and a set of states of the economy, the network parameters $\boldsymbol{\rho}$ are improved using variants of gradient descent, as described in section 4.2.2. In order to focus on obtaining a good approximation on the ergodic distribution of states of the economy, new training data is iteratively generated by simulating the economy, as described in section 4.2.3. Moreover, we continue to apply dense neural networks, as described in section 4.2.4, as function approximators. Following algorithm 1, we then put all ingredients together and minimize the loss function by iterating between simulating the economy, to obtain a new set of states $\mathcal{D}_{\text{train}}$, and performing a variant of gradient descent on $\mathcal{D}_{\text{train}}$.[23]

7.3 Performance of the algorithm

7.3.1 The setup

Economic parameters The labor endowment over the lifecycle remains the same as in the benchmark case (see fig. 2). Furthermore, the parameterization of the model is detailed table 5. Note that except for the additional parameters governing the collateral requirement for short-selling the bond and the convex adjustment costs of capital, the economic parameters remain identical to those in the benchmark model (see table 1). The convex adjustment costs on capital take the form $f(a_t^s, k_t^s) = \zeta\,(a_t^s - k_t^s)^2$, where $\zeta = 0.5$. The parameter κ, governing the collateral requirement for short selling the bond ($\kappa \cdot d_t^s + a_t^s \geq 0$) is taken to be $\kappa = 3$, which is high enough such that default does not occur in equilibrium. The bond promises a payout of 1 in the next period.

Architecture of the neural network We continue to use a densely connected neural network with two hidden layers, each with 1,000 nodes and relu activations. However, since the dimensionality of the state space almost doubles in the model, we aim to solve in this section in comparison to the benchmark case, the size of the input layer must be increased accordingly. That is, the input layer now includes not only the exogenous shock and the capital holding, but also the bond holding of each age-group. Since the deep equilibrium net approximates all equilibrium functions, *i.e.*, $2 \cdot (N-1)$ policy functions for the investment in capital and the corresponding KKT multipliers, $2 \cdot (N-1)$ policy functions for the investment in the one-period bond and corresponding KKT multipliers, as well as the price function for the price of the bond, the output layer now consists of $4 \cdot (N-1) + 1$ nodes.[24]

Hyper-parameters The hyper-parameters used for training the deep equilibrium net are summarized in table 6. In order to reduce the variance and size of the mini-batch gradient

[23] We monitor the level of the individual components of the loss function during training to ensure that each converges separately.

[24] Again, similar to the description in appendix B.1 for the benchmark case, we found it helpful to add additional but redundant information to the input layer. Furthermore, in order to retain a softplus activation in the output layer, we replaced the bond demand d_t^s in the equilibrium functions by $\kappa \cdot d_t^s + a_t^s$ (see, section 7.2.2, footnotes 20 and 21).

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 – 31,000	0.0001	100,000	1	1,000	1000 1000	relu relu
31,001 – 52,000	0.00001	100,000	1	10,000	1000 1000	relu relu

Table 6: Hyper-parameters chosen to train the deep equilibrium net for the benchmark model.

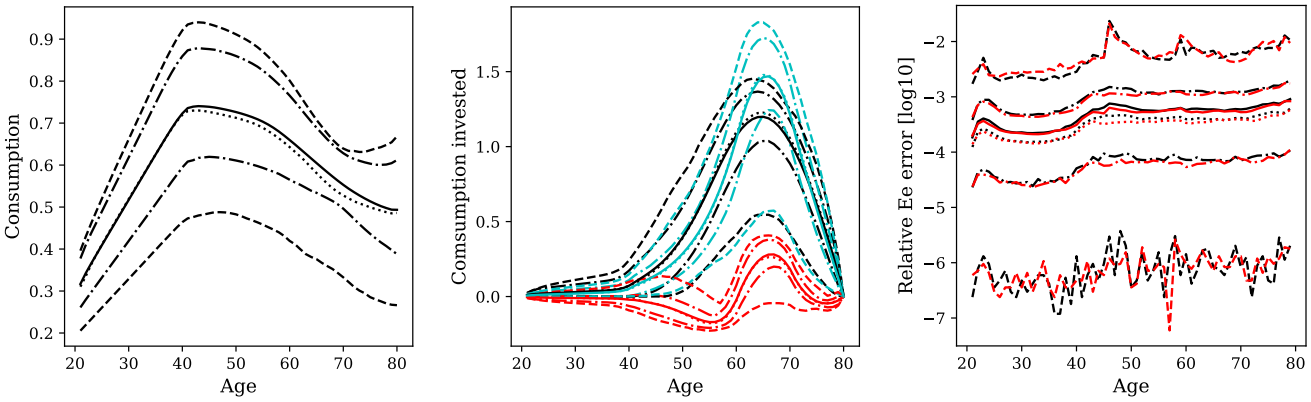


Figure 10: Learned equilibrium after 52,000 episodes of training. The left-hand side figure shows the consumption by age-group. The middle figure shows the investment in the one-period bond (red), in risky, illiquid capital (black), and overall (cyan). The right-hand side figure shows the relative errors in the Euler equations for the bond (red) and capital (black). All plots show the mean and several percentile values computed over 2,000 simulated periods. The dashed lines show the 0.1st and 99.9th percentile, the dash-dotted lines show the 10th and 90th percentile, the dotted line shows the median, and the solid line shows the mean.

descent steps, we decreased the learning rate and increased the mini-batch size after training the neural network on 31,000 episodes with 100,000 states each.

7.3.2 Training progress of the deep equilibrium net

We trained the neural network for 52,000 episodes, one epoch per episode, and 10^5 simulated periods per episode. In total, the neural network was, therefore, trained on more than 5×10^9 simulated states. After 31,000 episodes of training, we increase the mini-batch size from $m = 1{,}000$ to $m = 10{,}000$ and decreased the learning rate from $\alpha^{\text{learn}} = 0.0001$ to $\alpha^{\text{learn}} = 0.00001$. Lowering the learning rate towards the end of the training process decreases fluctuations in the loss function. Figure 10 shows the resulting consumption, investments, and relative Euler equation errors by age-group. The figure was generated by simulating 2,000 periods using the weights of the trained neural network (*i.e.*, after 52,000 episodes).

Table 7 reports the mean and maximum relative Euler equation (in %), KKT ($\times 10^{-2}$), and market clearing ($\times 10^{-2}$) errors as well as several percentiles on a simulated path of 10^4 periods, after training was finished.[25] As shown in the table, all mean errors are below 0.001 and the 99.9th percentile is always below 0.01.

[25] We simulated a total of 12,000 periods and discarded the first 2,000 periods.

	mean	max	0.1	10	50	90	99.9
Rel Ee capital [%]	0.05	2.97	0.00	0.01	0.03	0.11	0.64
Rel Ee bond [%]	0.05	4.25	0.00	0.00	0.03	0.10	0.67
KKT capital [$\times 10^{-2}$]	0.01	0.77	0.00	0.00	0.00	0.00	0.00
KKT bond [$\times 10^{-2}$]	0.03	1.04	0.00	0.00	0.00	0.02	0.18
Market clearing [$\times 10^{-2}$]	0.04	0.76	0.00	0.01	0.03	0.08	0.32
Market clearing (normalized) [$\times 10^{-2}$]	0.00	0.02	0.00	0.00	0.00	0.00	0.01

Table 7: 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 0.1, 10, 50, 90, and 99.9th percentile.

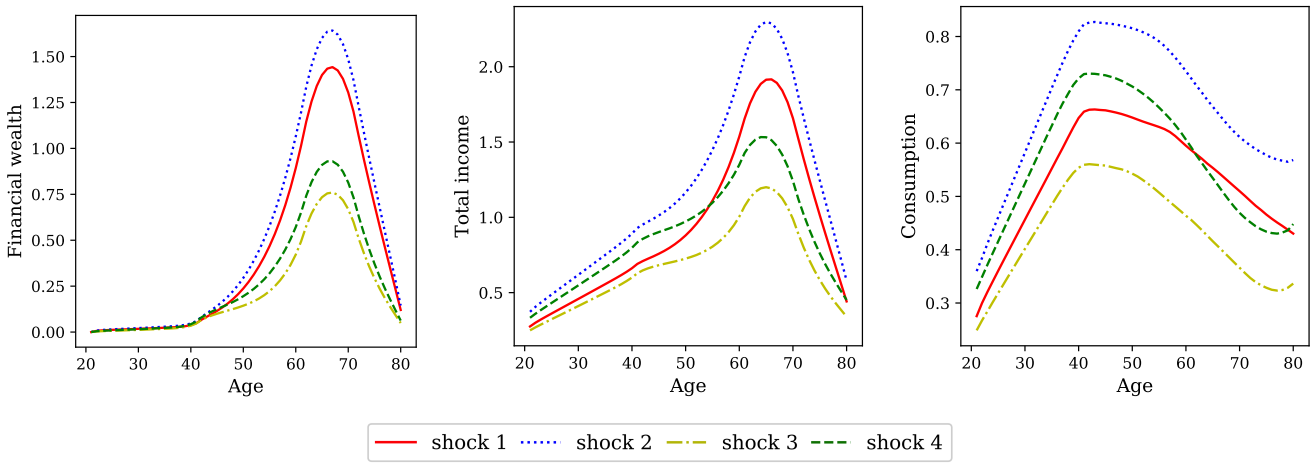


Figure 11: The first, second, and third figure displays the distribution of mean financial wealth (*i.e.*, cumulative value of capital and bond holdings), mean total income (*i.e.*, cash at hands), and mean consumption on a simulated path of 10,000 periods, respectively. The TFP and deprecation (η, δ) of shock 1, shock 2, shock 3, and shock 4 are (0.85, 0.5), (1.15, 0.5), (0.85, 0.9), and (1.15, 0.9), respectively.

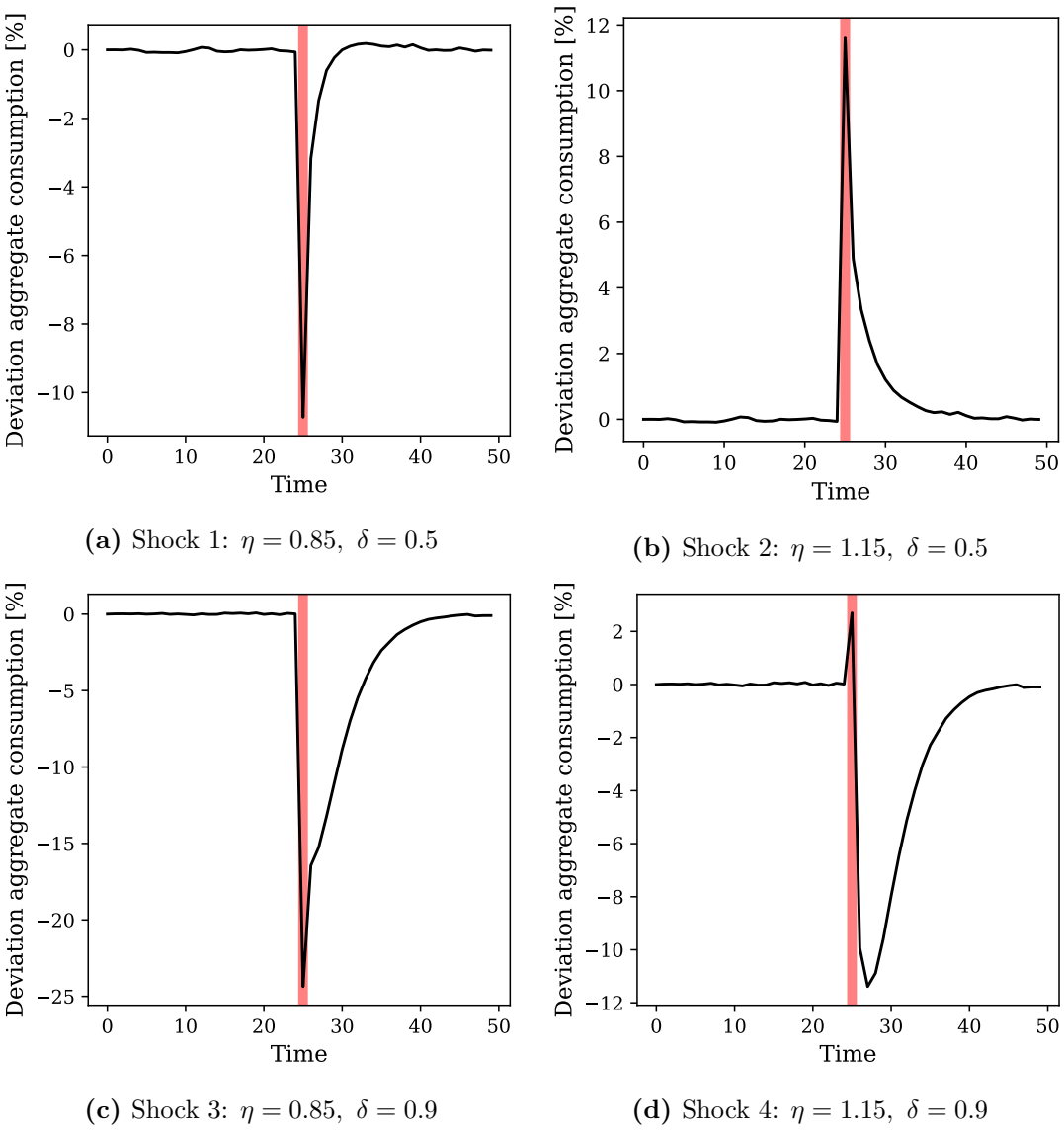
7.4 Economic results

In figure 11, we display the mean financial wealth, total income, and consumption across all age-groups contingent on the exogenous shock. The distributions resemble the case in which agents are not allowed to borrow (*cf.* the left column of fig. 7). This is due to the strict collateral requirements on the bond, which does not allow agents to be net-borrowers. Similar to the benchmark case with borrowing constraints, young agents cannot borrow against their future labor endowment and are often constrained.

Figure 12 shows the impulse response of the aggregate consumption to each of the four shocks.[26] Relative to the benchmark case with borrowing constraints (*cf.* fig. 8), the immediate reaction of the aggregate consumption is stronger for shocks 1, 2, and 3. Due to the convex adjustment costs on capital, agents generally consume less out of their savings in capital in reaction to a shock. The aggregate overall saving is equal to the saving in capital since the aggregate saving in the one-period bond is always zero due to market clearing. Because the aggregate saving in capital often reacts weaker in this model, aggregate consumption reacts stronger.

Figure 13 shows the cross-sectional reaction of consumption to each of the four shocks. As

[26]To obtain the impulse response, we simulate 50 time periods, once with a specific shock happening in period 25 and once without. We repeat this 200,000 times, compute the mean, and compare the simulations with the shock in period 25 and to those without. Again, we verified that the mean relative Euler equation error did not exceed 1%.



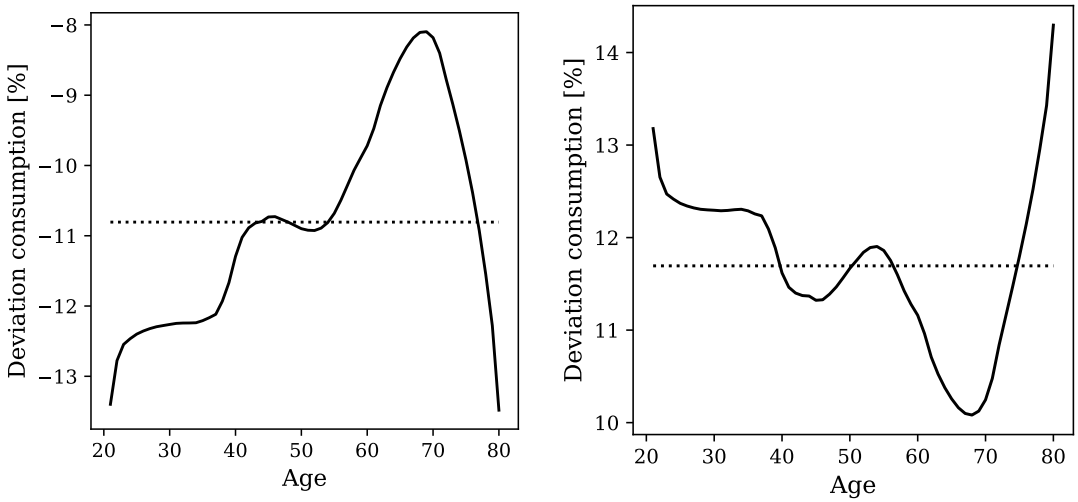
(a) Shock 1: $\eta = 0.85,\ \delta = 0.5$

(b) Shock 2: $\eta = 1.15,\ \delta = 0.5$

(c) Shock 3: $\eta = 0.85,\ \delta = 0.9$

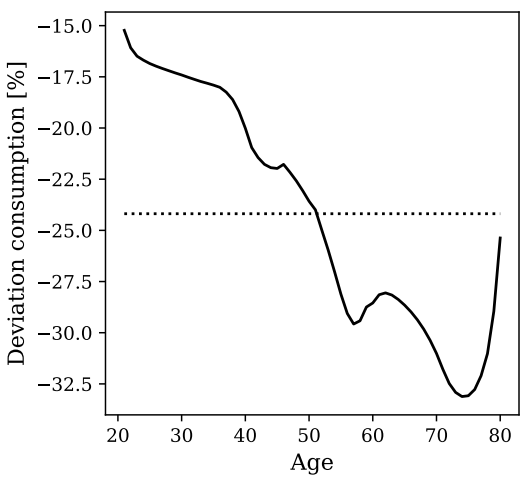
(d) Shock 4: $\eta = 1.15,\ \delta = 0.9$

Figure 12: Mean percent deviation of the aggregate consumption from its unconditional average when each of the exogenous shocks occurs in period 25. The mean impulse response was computed over 200,000 simulations. On impact of the respective shock, aggregate consumption change as follows: (a) decreases by 10.7%; (b) increases by 11.6%; (c) decreases by 24.4%; and (d) increases by 2.7%, respectively.

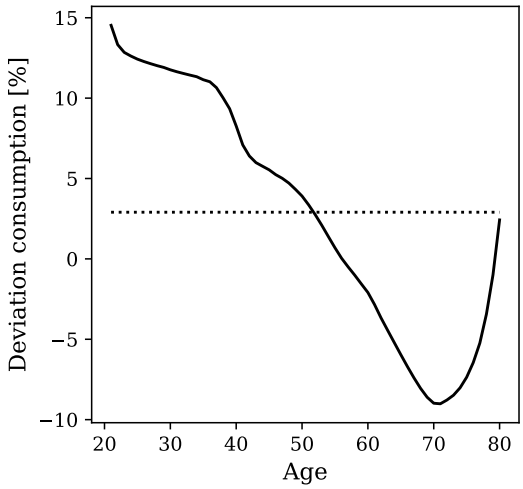


(a) Shock 1: $\eta = 0.85,\ \delta = 0.5$

(b) Shock 2: $\eta = 1.15,\ \delta = 0.5$



(c) Shock 3: $\eta = 0.85,\ \delta = 0.9$



(d) Shock 4: $\eta = 1.15,\ \delta = 0.9$

Figure 13: Mean percent deviation of consumption by age-group from its unconditional average when each of the exogenous shocks occurs. The mean is taken over 200,000 simulations.

the age-groups with large holdings of illiquid capital can consume less out of their savings, they adjust their consumption more strongly. Despite those with the largest holdings of illiquid capital also having positive bond holdings, their bond holdings are not enough to compensate for the illiquidity of majority of their wealth.

In the model presented in this section, the agents can choose between two assets to invest in illiquid, risky capital, and a liquid, risk-free bond. Therefore, this model allows us to study the impulse response of the interest rate and the equity premium. Figure 14 shows the impulse response of the return to capital, the interest rate, and the equity premium for each of the for shocks. The large TFP and depreciation shocks, together with the adjustment costs on capital, lead to a large equity premium of around 4%. We start by considering shocks 1 and 2, where depreciation is at its "normal" value and the TFP, which is i.i.d., is low (shock 1) or high (shock 2). On impact of shock 1, the lower TFP leads to a lower return on capital. This then decreases aggregate savings and hence aggregate capital in the following periods, which in turn leads to a higher return on capital. As the return of the bond is pinned down by the price in the previous period, it reacts with a one-period delay. The price of the bond drops on impact of shock 1 causing an increase of the bond return in the next period. Consequently, the equity premium drops on impact. One period after shock 1 has realized, the return on capital and on the bond are both higher than prior to the shock. As the bond return's deviation dominates, the equity premium is still below its unconditional average. This is due to the illiquidity of capital restricting agents' ability to withdraw from their savings in capital. Starting from the second period after shock 1, the equity premium exceeds its unconditional average. This is in line with the stylized fact of the equity premium being counter-cyclical (the two-period delay is due to the adjustment costs on capital). The analogue mechanism holds for shock 2 with opposite signs.

Similar workings can be observed for shock 3 in which the TFP is low, and depreciation is high. In contrast to shock 1, the persistence and magnitude of the high depreciation state leads to the adjustment costs playing an even more significant role. That is, rather than the return on capital reversing immediately in the period after the shock, it only reaches an above average value two periods later, and only peaks after four periods. The cross-sectional reaction to shock 4 (high depreciation and high TFP) is ambiguous. On impact, agents with little savings in capital, *i.e.*, young agents, may receive a higher income than on average and may, therefore, increase their savings. Agents with substantial savings in the capital, on the other hand, *i.e.*, agents around the retirement age, lose a large portion of their wealth and, therefore, reduce their holdings in the capital. As a result, the return on capital only reaches an above average value in the third period, and only peaks in the fifth period after the shock four has realized. The bond price adjusts immediately and, in direct consequence, the bond return reacts with a one-period delay. The equity premium reflects the interplay between these two mechanisms.

The results presented in this section indicate that it may be fruitful to jointly consider the interplay between age-groups, labor income, and portfolio decomposition across age-groups to study financial markets and the equity premium.

8 Conclusion

In this paper, we introduced deep equilibrium nets—neural networks that approximate all equilibrium functions directly 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 provide a grid-free, global, solution method, which can jointly address the

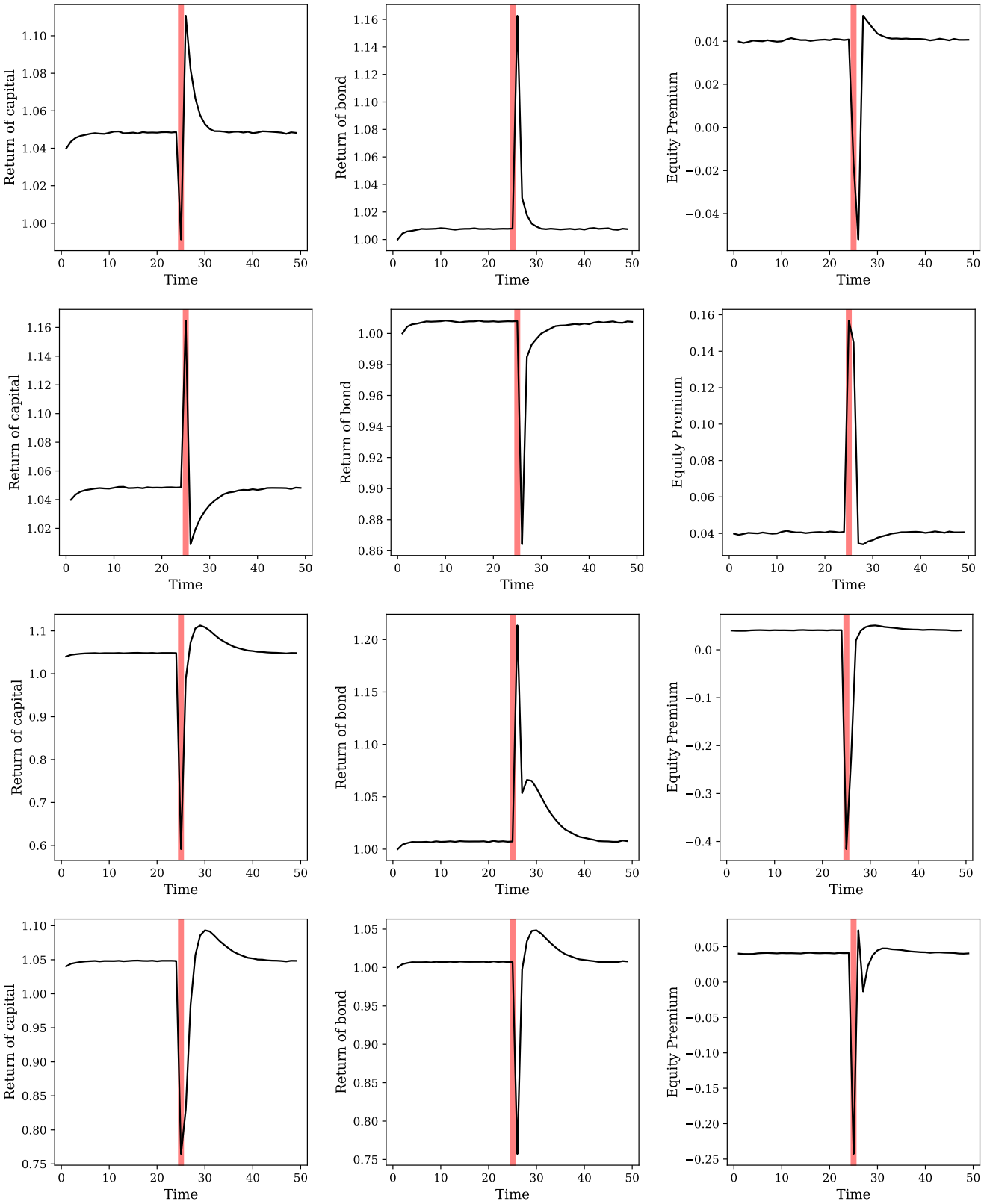


Figure 14: Mean return on capital (left), the one-period bond (middle), and the equity premium (*i.e.*, the return of capital minus the return of the bond, right). The first, second, third, and fourth row shows the reaction to shock 1, shock 2, shock 3, and shock 4 occurring in period 25, respectively. The mean is computed over 200,000 simulations.

computational challenges arising from a high-dimensional state space, significant uncertainty, financial frictions, and irregular geometries of the state space. We illustrate the performance of our method for two models with a 59 and 117-dimensional endogenous state space, occasionally binding constraints, and aggregate uncertainty. For both models, we obtain mean relative errors in the Euler equations of order 10^{-4}.

Next, we study how the level of exogenous borrowing constraints affects the consumption response to aggregate shocks across age-groups. We find that the consumption response is heterogeneous across age-groups and strongly dependent on the level of the constraint, underlining the need for macro-economic models and consequently, solution methods, which can address both micro-level heterogeneity and financial frictions. We then enrich the model by allowing agents to invest in two types of assets: risky, illiquid capital and a risk-free, one-period bond subject to collateral requirements. We study how this change affects the consumption response as well as the response of risky return, the risk-free rate, and the equity premium to aggregate shocks. The model generates a mean equity premium of around 4%. Our findings suggest that it may be fruitful to study the heterogeneous response to aggregate shocks across age-groups to understand the evolution of asset returns and the equity premium.

References

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.

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.

Cai, Y., Judd, K., and Steinbuks, J. (2017). A nonlinear certainty equivalent approximation method for dynamic stochastic problems. *Quantitative Economics*, 8(1):117–147.

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

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.

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

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.

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.

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*.

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

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. (2019). Making carbon taxation a generational win win. Working paper.

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.

Krueger, D., Perri, F., Pistaferri, L., and Violante, G. L. (2010). Cross-sectional facts for macroeconomists. *Review of Economic Dynamics*, 13(1):1 – 14. Special issue: Cross-Sectional Facts for Macroeconomists.

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.

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.

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.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition.

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.

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

Saez, E. and Zucman, G. (2016). Wealth Inequality in the United States since 1913: Evidence from Capitalized Income Tax Data *. *The Quarterly Journal of Economics*, 131(2):519–578.

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*.

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.

Villa, A. T. and Valaitis, V. (2018). Machine learning projection methods for macro-finance models. Working paper.

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

Zucman, G. (2019). Global wealth inequality. *Annual Review of Economics*, 11.

Appendices

A Background on neural networks

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

Gradient descent is a popular 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. When working with large datasets, gradient descent is computationally expensive. That is, optimizing a neural network's parameters requires many gradient update steps (upwards of many thousands). Computing the gradients with the full dataset is very time intensive. There are various tools and techniques available to reduce training time.

The learning rate is one hyper-parameter that can be tuned to reduce the number of steps needed to reach the optimum. Large learning rates correspond to taking larger steps relative to the gradient (see eq. (39)). However, by choosing a learning rate that is too large, we can overshoot and oscillate around the optimum and never reach the desired performance. Additionally, specific to our application, large learning rates can lead to an over-correction, pushing the parameters to a region where they generate economically nonsensical values (see appendix B.2). Choosing a learning rate that is too small will increase the duration of training. Furthermore, the learning rate can be altered during training. We found it helpful to use a larger learning rate at the beginning of training, and decrease it for the final episodes in the bond case. In general, the range in which we could tune the learning rate was limited by the issue discussed in appendix B.2.

Mini-batch stochastic gradient descent is another method that can be used. This method allows the modeler to get a gradient update at a lower computational cost by estimating the gradient on a subset of randomly sampled data (without replacement). The subset of the data is referred to as a *mini-batch*. An epoch has been completed once all of the data points have been sampled for a mini-batch. Different mini-batch sizes have different trade-offs. As already discussed, using the full dataset is computationally expensive as the whole dataset is needed for the computation for a single update. However, the gradients computed are not estimates and are in the true descending direction. At the other extreme, a mini-batch size of 1 will be computationally cheap. Per epoch, N updates are performed (N = number of data points).

However, the estimates of the gradient are very noisy and have high variance leading to steps that may be in a false direction. Increasing the mini-batch size will increase the quality of the estimate (reducing the variance) and only slowly increases the computational cost. That is, as a mini-batch size of 1 underutilizes the computational resources, larger mini-batch sizes (that are still on the low end of the spectrum) can be computed at virtually no additional cost. This is especially true when utilizing graphics processing units that can process many computational steps in parallel.

Additional methods have been developed to attempt to make the performance of neural networks more consistent. Momentum adds a fraction of the weighted average of the previous gradients to the current gradient. Adaptive methods adapt the learning rate to the parameters by normalizing the learning rate by a weighted average of the gradient's second moment. Adam (Kingma and Ba, 2014) uses both decayed momentum and a decayed adaptive learning rate when making updates. The reader is referred to Ruder (2016) for a more extensive overview of neural network optimization methods.

B Practical issues

B.1 Passing redundant information to the equilibrium net

Heuristically, we found it to be helpful to pass redundant information to the neural network in order 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 heuristically fount 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 $\mathbf{f}$, the labor endowment across age-groups $\mathbf{l}(z)$, the distribution of financial wealth $\mathbf{i}_{\text{fin}} = r \cdot \mathbf{k}$, the distribution of labor income $\mathbf{i}_{\text{labor}} = w \cdot \mathbf{l}(z)$, the distribution of cash at hands $\mathbf{i}_{\text{total}} = \mathbf{i}_{\text{labor}} + \mathbf{i}_{\text{fin}}$, and the transition probabilities to the state tomorrow $\Pi[z, :]$. To summarize, instead of passing a $N+1$ dimensional vector $\mathbf{x}^{\text{sufficient}} = [z,\ \mathbf{k}^T]^T$ to the neural network, we pass a $5 \cdot N + 12$ dimensional vector $\mathbf{x}^{\text{redundant}} = [z,\ \eta(z),\ \delta(z),\ K,\ L,\ r,\ w,\ Y,\ \mathbf{k}^T,\ \mathbf{i}_{\text{fin}}^T,\ \mathbf{l}^T,\ \mathbf{i}_{\text{labor}}^T,\ \mathbf{i}_{\text{total}}^T,\ \Pi[z, 1:4]]^T$ to the neural network.

B.2 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 not necessary anymore. 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. 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 consumptions 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 networks predict a policy implying negative consumption, the optimality conditions can still be evaluated, while the large punishment term will guide the parameter updates so that improved policies do not imply negative consumption. Similarly for aggregate capital:

- set a punishment parameter: $\epsilon^{\text{punish}} = 10^{-5}$.
- if the predicted aggregate capital is less than ϵ^{punish}, replace by ϵ^{punish}: $\hat{K}_{\text{adjusted}}(\mathbf{x}) = \max(\hat{K}(\mathbf{x}), \epsilon^{\text{punish}})$.
- add the punishment term $\left(\frac{1}{\epsilon^{\text{punish}}} \max(-\hat{K}(\mathbf{x}), 0)\right)^2$ to the loss function.