Solving Life-Cycle Models with a Rich Asset Structure using Deep Learning

Marlon Azinović-Yang[1]

based on joint work with Jan Žemlička,[2,3] Luca Gaegauf,[2] and Simon Scheidegger[4,5]

[1]University of Pennsylvania

[2]University of Zurich, [3]Swiss Finance Institute, [4]University of Lausanne, [5]E4S

SUERF, ECB, Bank of Finland and Bank of Italy Workshop on the Use of AI in Economic Modelling and Forecasting

June 26, 2024

Motivation

- Economic models to study questions related to aggregate risk and asset pricing, often require global solution methods to compute equilibria
- Computing a functional rational expectations equilibrium amounts to computing a set of functions, f_i, mapping the state of the economy, $\mathbf{x}$, to endogenous outcomes $f_i(\mathbf{x})$:

$$f_i : \mathcal{D} \subset \mathbb{R}^{N_{\text{in}}} \to \mathbb{R} : \underbrace{\mathbf{x}}_{\text{state}} \to \underbrace{f_i(\mathbf{x})}_{\text{endogenous variables}}, \text{ s.t. : } \underbrace{\mathbf{G}(\mathbf{x}, f_1, \ldots, f_{N_{\text{out}}}) = 0}_{\text{equilibrium conditions}}$$

- This can be a computationally demanding task, especially when
  - the state of the economy is high-dimensional
  - the equilibrium functions are nonlinear
- Both often happens for Overlapping Generations (OLG) models:
  - the state includes the wealth distribution across age-groups
  - young households are often constrained
  - may want to account for portfolio decomposition and volatility of labor income, both of which have strong lifecycle components

This talk

- Basic solution method developed in Azinovic et al. (2022)
- More recent progress on portfolio choice and market clearing neural network architectures developed in Azinovic and Žemlička (2023)

This talk

- Basic solution method developed in Azinovic et al. (2022)
- More recent progress on portfolio choice and market clearing neural network architectures developed in Azinovic and Žemlička (2023)
- Other papers on deep learning based solution methods I learned a lot form: Maliar et al. (2021); Kase et al. (2023); Gu et al. (2023); Kahou et al. (2021); Han et al. (2022); Valaitis and Villa (2024); Kahou et al. (2022); Fernández-Villaverde et al. (2023); Barnett et al. (2023); Jungerman (2023); Kahou et al. (2024)

Deep Equilibrium Nets

Violations of equilibrium conditions as loss function

Basic idea in Azinovic et al. (2022): write equilibrium conditions as

$$\mathbf{G}(\mathbf{x}, \mathbf{f}) = 0 \ \forall \mathbf{x}$$

$$\begin{aligned} \mathbf{G} &: \text{ equilibrium conditions: FOC's, market clearing, Bellman equations, } \ldots \\ \mathbf{x} &: \text{ state of the economy} \\ \mathbf{f} &: \text{ equilibrium functions.} \end{aligned}$$

Approximate $\mathbf{f}$ by neural network $\mathcal{N}_\rho$

$$\mathcal{N}_\rho(\mathbf{x}) \approx \mathbf{f}(\mathbf{x})$$

How?

Violations of equilibrium conditions as loss function

Basic idea in Azinovic et al. (2022): write equilibrium conditions as

$$\mathbf{G}(\mathbf{x}, \mathbf{f}) = 0 \ \forall \mathbf{x}$$

$$\begin{aligned} \mathbf{G} &: \text{ equilibrium conditions: FOC's, market clearing, Bellman equations, } \dots \\ \mathbf{x} &: \text{ state of the economy} \\ \mathbf{f} &: \text{ equilibrium functions.} \end{aligned}$$

Approximate $\mathbf{f}$ by neural network $\mathcal{N}_{\boldsymbol{\rho}}$

$$\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x})$$

How?

Standard deep learning:

- need labeled data, i.e. inputs for which we know the true output: $\{\mathbf{x}_i, f(\mathbf{x}_i)\}_i$
- train neural network parameters $\boldsymbol{\rho}$ to minimize the loss function

$$\ell_{\boldsymbol{\rho}} := \frac{1}{N_{\text{labeled data}}} \sum_{\mathbf{x}_i} \left(f(\mathbf{x}_i) - \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}_i)\right)^2$$

Violations of equilibrium conditions as loss function

Basic idea in Azinovic et al. (2022): write equilibrium conditions as

$$\mathbf{G}(\mathbf{x}, \mathbf{f}) = 0 \ \forall \mathbf{x}$$

$$\begin{aligned} \mathbf{G} &: \text{ equilibrium conditions: FOC's, market clearing, Bellman equations, } \ldots \\ \mathbf{x} &: \text{ state of the economy} \\ \mathbf{f} &: \text{ equilibrium functions.} \end{aligned}$$

Approximate $\mathbf{f}$ by neural network $\mathcal{N}_{\boldsymbol{\rho}}$

$$\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x})$$

How?

Standard deep learning:

- need labeled data, i.e. inputs for which we know the true output: $\{\mathbf{x}_i, f(\mathbf{x}_i)\}_i$
- train neural network parameters $\boldsymbol{\rho}$ to minimize the loss function

$$\ell_{\boldsymbol{\rho}} := \frac{1}{N_{\text{labeled data}}} \sum_{\mathbf{x}_i} \left(f(\mathbf{x}_i) - \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}_i)\right)^2$$

Deep equilibrium nets:

- use equilibrium conditions directly as loss function

$$\ell_{\boldsymbol{\rho}} := \frac{1}{N_{\text{path length}}} \sum_{\mathbf{x}_i \text{ on sim. path}} \left(\mathbf{G}(\mathbf{x}_i, \mathcal{N}_{\boldsymbol{\rho}})\right)^2$$

- no need for labeled data! What are Neural Nets? Why use Neural Nets?

Training DEQNs

1. Simulate a sequence of states $\mathcal{D}^i_{\text{train}} \leftarrow \{\mathbf{x}^i_1, \mathbf{x}^i_2, \ldots, \mathbf{x}^i_T\}$ from the policy encoded by the network parameters $\boldsymbol{\rho}^i$.
2. Evaluate the errors of the equilibrium conditions on the newly generated set $\mathcal{D}_{\text{train}}$.
3. If the error statistics are not low enough:
   3.1 update the parameters of the neural network with a gradient descent step (or a variant):
   $$\rho^{i+1}_k = \rho^i_k - \alpha_{\text{learn}} \frac{\partial \ell_{\mathcal{D}^i_{\text{train}}}(\boldsymbol{\rho}^i)}{\partial \rho^i_k}.$$
   3.2 set new starting states for simulation: $\mathbf{x}^{i+1}_0 = \mathbf{x}^i_T$.
   3.3 increase i by one and go back to step 1.

Illustrative Model

Illustrative OLG model with capital and bond

- Representative firm produces with

$$
\begin{aligned}
F(z_t, K_t, L) &= z_t K_t^{\alpha} L^{1-\alpha} \\
w_t &= \alpha z_t K_t^{\alpha-1} L^{1-\alpha} \\
r_t &= z_t (1-\alpha) K_t^{\alpha} L^{\alpha}
\end{aligned}
$$

- Uncertainty in TFP z_t, and depreciation of capital δ_t

$$
\begin{aligned}
\log(z_{t+1}) &= \rho_z \log(z_t) + \sigma_z \epsilon_t \\
\epsilon_t &\sim N(0,1) \\
\delta_t &= \delta \frac{2}{1+z}
\end{aligned}
$$

- Assets
  - one period bond with price p_t in aggregate supply B
  - risky capital K_t
  - borrowing constraints on both assets

$$
\begin{aligned}
b_t^h &\geq 0 \\
k_t^h &\geq 0
\end{aligned}
$$

- Households
  - $H = 32$ age-groups, indexed with $h \in \mathcal{H} := \{1, \ldots, 32\}$
  - supply labor units l_t^h inelastically
  - adjustment costs on capital

$$
\begin{aligned}
\Delta_{k,t}^h &:= k_{t+1}^{h+1} - k_t^h \\
\text{adj. costs} &= \psi \left(\Delta_{k,t}^h \right)^2
\end{aligned}
$$

  - budget constraint

$$
\begin{aligned}
c_t^h = {} & l^h w_t + b_{t-1}^{h-1} + k_{t-1}^{h-1}(1 - \delta_t + r_t) \\
& - p_t^b b_t^h - k_t^h - \psi \left(\Delta_{k,t}^h \right)^2
\end{aligned}
$$

  - maximize

$$
\mathrm{E}\left[\sum_{i=h}^{H} \beta^{i-h} u(c_{t+i}^{h+i}) \right]
$$

$$
u(c) := \frac{c^{1-\gamma} - 1}{1-\gamma}
$$

Equilibrium conditions

- **Market clearing**:

$$K_t := \sum_{h \in \mathcal{H}} k_t^h$$

$$B = \sum_{h \in \mathcal{H}} b_t^h \Leftrightarrow \epsilon_t^B := B - \sum_{h \in \mathcal{H}} b_t^h = 0$$

- **Firms optimize**:

$$w_t := \alpha z_t K_t^{\alpha - 1} L^{1-\alpha}$$

$$r_t := z_t (1-\alpha) K_t^{\alpha} L^{\alpha}$$

- **Households optimize**:
  - H sets of Karush Kuhn Tucker conditions for bond
    $\Rightarrow$ single equation using the Fisher-Burmeister equation
    $\Rightarrow$ H errors $\epsilon_t^{k,i}$
  - H sets of Karush Kuhn Tucker conditions for capital
    $\Rightarrow$ single equation using the Fisher-Burmeister equation
    $\Rightarrow$ H errors $\epsilon_t^{h,i}$

details

Approximation with standard DEQN

- State of the economy

$$\mathbf{x}_t = [\underbrace{z_t}_{\text{ex. shock}}, \underbrace{k_t^1, \dots, k_t^{32}}_{\text{dist. of cap.}}, \underbrace{b_t^1, \dots, b_t^{32}}_{\text{dist. of bonds}}]$$

- Equilibrium policies

$$\mathbf{f}(\mathbf{x}_t) = [\underbrace{k_{t+1}^1, \dots, k_{t+1}^{32}}_{\text{capital policy}}, \underbrace{b_{t+1}^1, \dots, b_{t+1}^{32}}_{\text{bond policy}}, \underbrace{p_t^b}_{\text{bond price}}]$$

- Neural network approximates

$$\mathcal{N}_\rho(\mathbf{x}_t) = [\underbrace{\hat{k}_{t+1}^1, \dots, \hat{k}_{t+1}^{32}}_{\text{capital policy}}, \underbrace{\hat{b}_{t+1}^1, \dots, \hat{b}_{t+1}^{32}}_{\text{bond policy}}, \underbrace{\hat{p}_t^b}_{\text{bond price}}] \approx \mathbf{f}(\mathbf{x}_t)$$

- Loss function

$$\ell_\rho(\mathbf{x}_t) := \underbrace{w_{hh,k}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{k,h}\right)^2\right)}_{\text{opt. cond. cap.}} + \underbrace{w_{hh,b}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{b,h}\right)^2\right)}_{\text{opt. cond. bond}} + \underbrace{w_{mc,B}}_{\text{weight}} \underbrace{\left(\epsilon_t^{B}\right)^2}_{\text{market clearing}}$$

Innovation 1: Market clearing layers

- Neural network first predicts

$$\mathcal{N}^{\text{pre}}_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, \tilde{b}^1_{t+1}, \ldots, \tilde{b}^{32}_{t+1}, \hat{p}^b_t]$$

- Apply transformation $m(\ldots, \cdot)$

$$[\hat{b}^1_{t+1}, \ldots, \hat{b}^{32}_{t+1}] = m\left(\mathcal{N}^{\text{pre}}_{\boldsymbol{\rho}}(\mathbf{x}_t), B\right)$$

- Such that

$$B = \sum_{h=1}^{32} \hat{b}^h_{t+1}$$

- Put together

$$\mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}_t) := [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, \hat{b}^1_{t+1}, \ldots, \hat{b}^{32}_{t+1}, \hat{p}^b_t]$$

- Loss function now

$$\ell_{\boldsymbol{\rho}}(\mathbf{x}_t) := \underbrace{w_{hh,k}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon^{k,h}_t\right)^2 \right)}_{\text{opt. cond. cap.}} + \underbrace{w_{hh,b}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon^{b,h}_t\right)^2 \right)}_{\text{opt. cond. bond}} + \cancelto{=0}{\underbrace{w_{mc,B}}_{\text{weight}} \underbrace{\left(\epsilon^{B}_t\right)^2}_{\text{market clearing}}}$$

Innovation 1: Market clearing layers

- Neural network first predicts

$$\mathcal{N}_{\rho}^{\text{pre}}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, \tilde{b}_{t+1}^{1}, \ldots, \tilde{b}_{t+1}^{32}, \hat{p}_t^{b}]$$

- Apply transformation $m(\ldots, \cdot)$

$$[\hat{b}_{t+1}^{1}, \ldots, \hat{b}_{t+1}^{32}] = m\left(\mathcal{N}_{\rho}^{\text{pre}}(\mathbf{x}_t), B\right)$$

- Such that

$$B = \sum_{h=1}^{32} \hat{b}_{t+1}^{h}$$

- Put together

$$\mathcal{N}_{\rho}(\mathbf{x}_t) := [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, \hat{b}_{t+1}^{1}, \ldots, \hat{b}_{t+1}^{32}, \hat{p}_t^{b}]$$

- Loss function now

$$\ell_{\rho}(\mathbf{x}_t) := \underbrace{w_{hh,k}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{k,h}\right)^2\right)}_{\text{opt. cond. cap.}} + \underbrace{w_{hh,b}}_{\text{weight}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{b,h}\right)^2\right)}_{\text{opt. cond. bond}} + \cancelto{0}{\underbrace{w_{mc,B}}_{\text{weight}} \underbrace{\left(\epsilon_t^{B}\right)^2}_{\text{market clearing}}}$$

1. no need to learn economics we already know ex-ante
2. remaining loss easier to interpret
3. states simulated from the policy are always consistent with market clearing ▸ details

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N - 1$ asset models are nested in N asset models

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}_{\rho}^{1}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, 0\times\hat{b}_{t+1}^{1}, \ldots, 0\times\hat{b}_{t+1}^{32}, \hat{p}_t^{b}], B^{1} = 0$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}^1_\rho(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0\times\hat{b}^1_{t+1}, \ldots, 0\times\hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}^1_{\rho}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \dots, \hat{k}^{32}_{t+1}, 0 \times \hat{b}^1_{t+1}, \dots, 0 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}_{\boldsymbol{\rho}}^{1}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, 0\times\hat{b}_{t+1}^{1}, \ldots, 0\times\hat{b}_{t+1}^{32}, \hat{p}_t^{b}], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}_{\boldsymbol{\rho}}^{2}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, 1\times\hat{b}_{t+1}^{1}, \ldots, 1\times\hat{b}_{t+1}^{32}, \hat{p}_t^{b}], B^2 = 0.1$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}_{\boldsymbol{\rho}}^{1}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, 0 \times \hat{b}_{t+1}^{1}, \ldots, 0 \times \hat{b}_{t+1}^{32}, \hat{p}_t^b], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}_{\boldsymbol{\rho}}^{3}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, 1 \times \hat{b}_{t+1}^{1}, \ldots, 1 \times \hat{b}_{t+1}^{32}, \hat{p}_t^b], B^3 = 0.2$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}^1_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0\times\hat{b}^1_{t+1}, \ldots, 0\times\hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}^4_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 1\times\hat{b}^1_{t+1}, \ldots, 1\times\hat{b}^{32}_{t+1}, \hat{p}^b_t], B^4 = 0.3$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}^1_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0 \times \hat{b}^1_{t+1}, \ldots, 0 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}^5_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 1 \times \hat{b}^1_{t+1}, \ldots, 1 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^5 = 0.4$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}^1_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0 \times \hat{b}^1_{t+1}, \ldots, 0 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}^{\cdots}_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 1 \times \hat{b}^1_{t+1}, \ldots, 1 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^{\cdots} = \ldots$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}_{\boldsymbol{\rho}}^{1}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0 \times \hat{b}^1_{t+1}, \ldots, 0 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}_{\boldsymbol{\rho}}^{100}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 1 \times \hat{b}^1_{t+1}, \ldots, 1 \times \hat{b}^{32}_{t+1}, \hat{p}^b_t], B^{100} = 10$$

Innovation 2: Stabilizing step-wise model transformations

- Single asset models are easy
- Many asset models are hard
- Why?
  - portfolio choice only pinned down at low errors in equilibrium conditions
  - but how do we get there?
- Step-wise model transformations
  1. $N-1$ asset models are nested in N asset models
  2. start with single asset model

$$\mathcal{N}_{\boldsymbol{\rho}}^{1}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0\times\hat{b}^1_{t+1}, \ldots, 0\times\hat{b}^{32}_{t+1}, \hat{p}^b_t], B^1 = 0$$

  3. solve the model
  4. train the neural network to predict the bond price (supervised, from zero liquidity limit)
  5. slowly introduce the second asset (such that the error remains low)

$$\mathcal{N}_{\boldsymbol{\rho}}^{100}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 1\times\hat{b}^1_{t+1}, \ldots, 1\times\hat{b}^{32}_{t+1}, \hat{p}^b_t], B^{100} = 10$$

  6. equilibrium errors always remain low

Application

Step 1: Solve single asset model

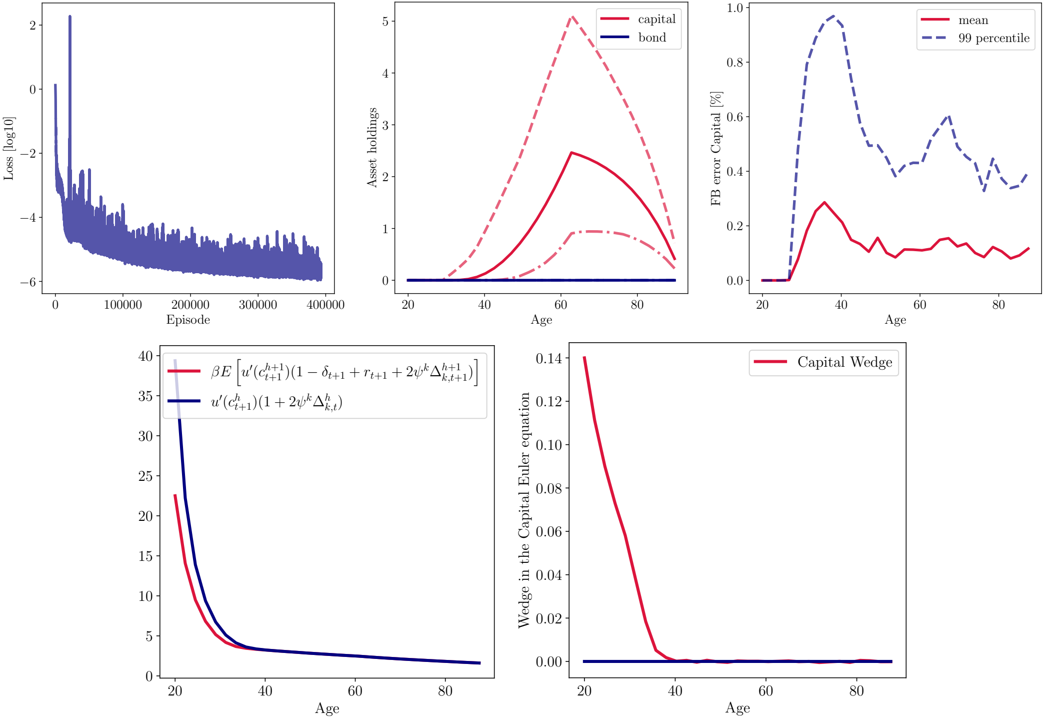
- Borrowing constraint $\underline{b} = 0$, net-supply $B = 0$
- Neural network predicts

$$\mathcal{N}_{\boldsymbol{\rho}}^{\text{pre}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0 \times \tilde{b}^1_{t+1}, \ldots, 0 \times \tilde{b}^{32}_{t+1}, \hat{p}^b_t]$$

$$\Rightarrow \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0, \ldots, 0, \hat{p}^b_t]$$

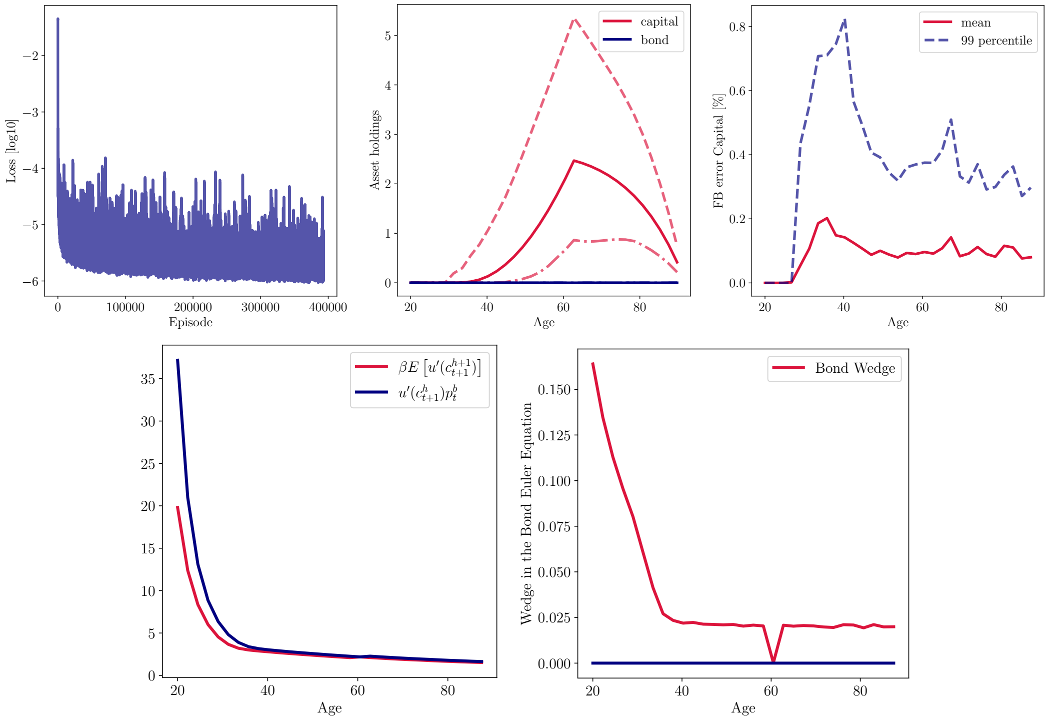
- Loss function

$$\ell_{\boldsymbol{\rho}}(\mathbf{x}_t) := 1 \times \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{k,h} \right)^2 \right)}_{\text{opt. cond. cap.}} + \underbrace{0 \times \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{b,h} \right)^2 \right)}_{\text{opt. cond. bond}}}_{=0}$$


Loss [log10]
Episode
Asset holdings
capital
bond
Age
FB error Capital [%]
mean
99 percentile
$\beta E\left[u'(c_{t+1}^{h+1})(1-\delta_{t+1}+r_{t+1}+2\psi^k\Delta_{k,t+1}^{h+1})\right]$
$u'(c_{t+1}^{h})(1+2\psi^k\Delta_{k,t}^{h})$
Wedge in the Capital Euler equation
Capital Wedge

Step 2: Pre-train bond price in the capital only model

- Keep borrowing constraint $\underline{b} = 0$, net-supply $B = 0$, and neural network masks
$$\mathcal{N}_\rho(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, 0, \ldots, 0, \hat{p}^b_t]$$
- In equilibrium we know that
$$p^b_t \geq \frac{\beta \mathrm{E}\left[u'(c^{h+1}_{t+1})\right]}{u'(c^h_t)}$$
with equality for unconstrained agents.
- With market clearing policies, we have a closed form expression for the bond price and can define pre-train price and error
$$\begin{aligned} p^{b,\text{pre-train}}_t &:= \max_{h \in \mathcal{H}} \left\{ \frac{\beta \mathrm{E}\left[u'(c^{h+1}_{t+1})\right]}{u'(c^h_t)} \right\} \\ \epsilon^{\text{pre-train}}_t &:= p^{b,\text{pre-train}}_t - \hat{p}^b_t \end{aligned}$$
- Loss function
$$\ell_\rho(\mathbf{x}_t) := 1 \times \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon^{k,h}_t\right)^2 \right)}_{\text{opt. cond. cap.}} + 0 \times \underbrace{\underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon^{b,h}_t\right)^2 \right)}_{\text{opt. cond. bond}}}_{=0} + 1 \times \underbrace{\left(\epsilon^{\text{pre-train}}_t\right)^2}_{\substack{\text{price pre-train error} \\ \text{train supervised}}}$$


Loss [log10]
Episode
Asset holdings
capital
bond
Age
FB error Capital [%]
mean
99 percentile
Age
$\beta E\left[u'(c_{t+1}^{h+1})\right]$
$u'(c_{t+1}^{h})p_t^b$
Age
Wedge in the Bond Euler Equation
Bond Wedge
Age

Step 3: Slowly increase bond supply

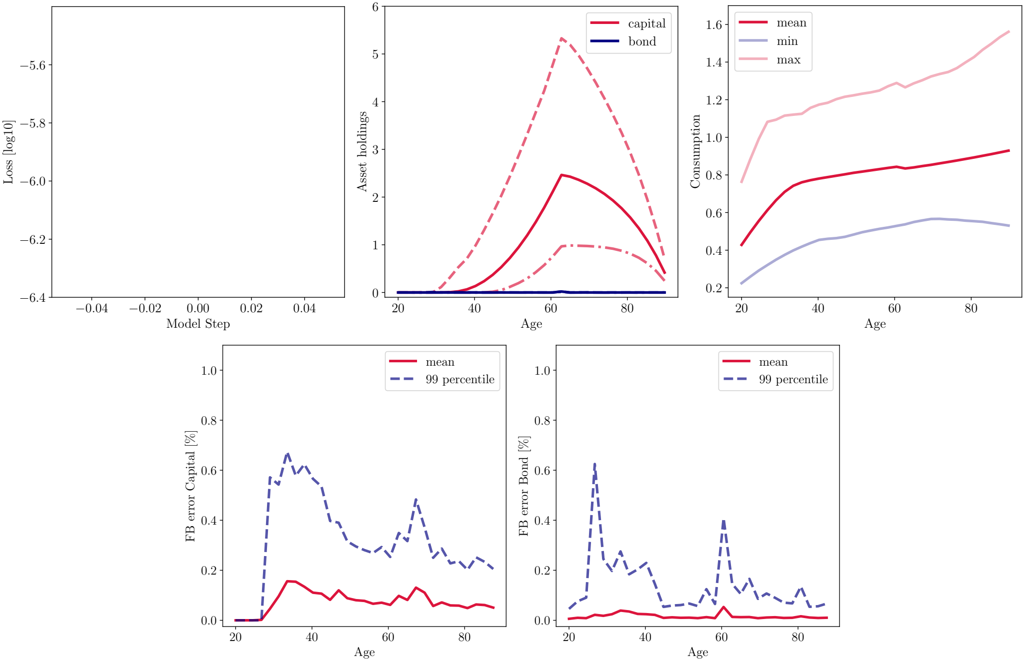
- Borrowing constraint $\underline{b} = 0$, increase net-supply from $B = 0.1$ to $B = 10$
- Neural network predicts

$$\mathcal{N}_{\boldsymbol{\rho}}^{\text{pre}}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, \underbrace{0.01 \times \tilde{b}_{t+1}^{1}, \ldots, 0.01 \times \tilde{b}_{t+1}^{32}}_{\text{bond policies active}}, \hat{p}_t^b]$$

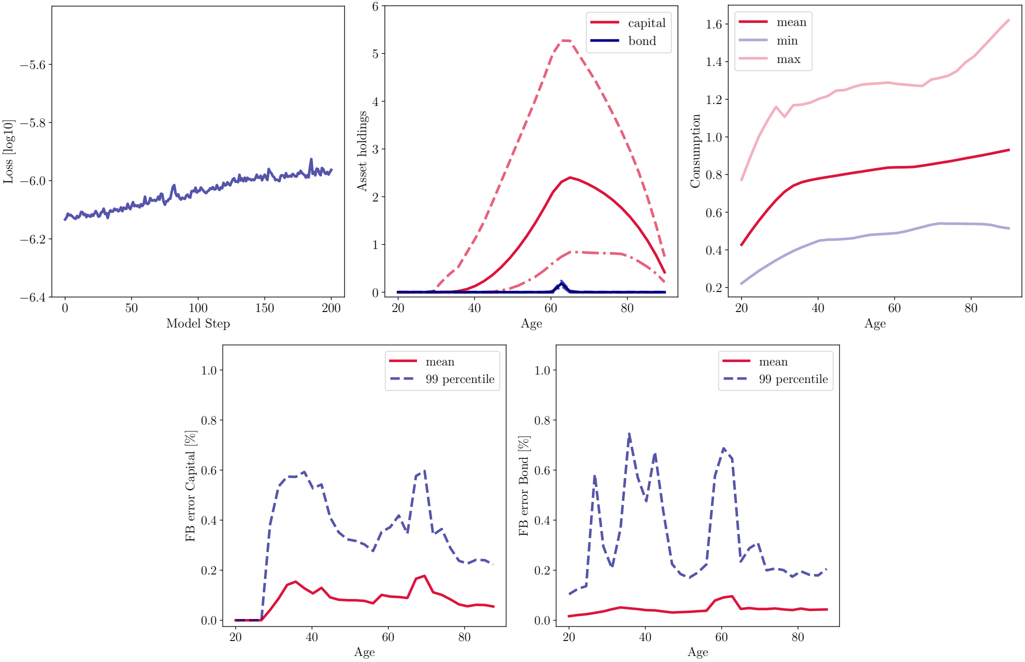
$$\Rightarrow \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}_t) = [\hat{k}_{t+1}^{1}, \ldots, \hat{k}_{t+1}^{32}, \underbrace{\hat{b}_{t+1}^{1}, \ldots, \hat{b}_{t+1}^{32}}_{\text{always add up the B}}, \hat{p}_t^b]$$

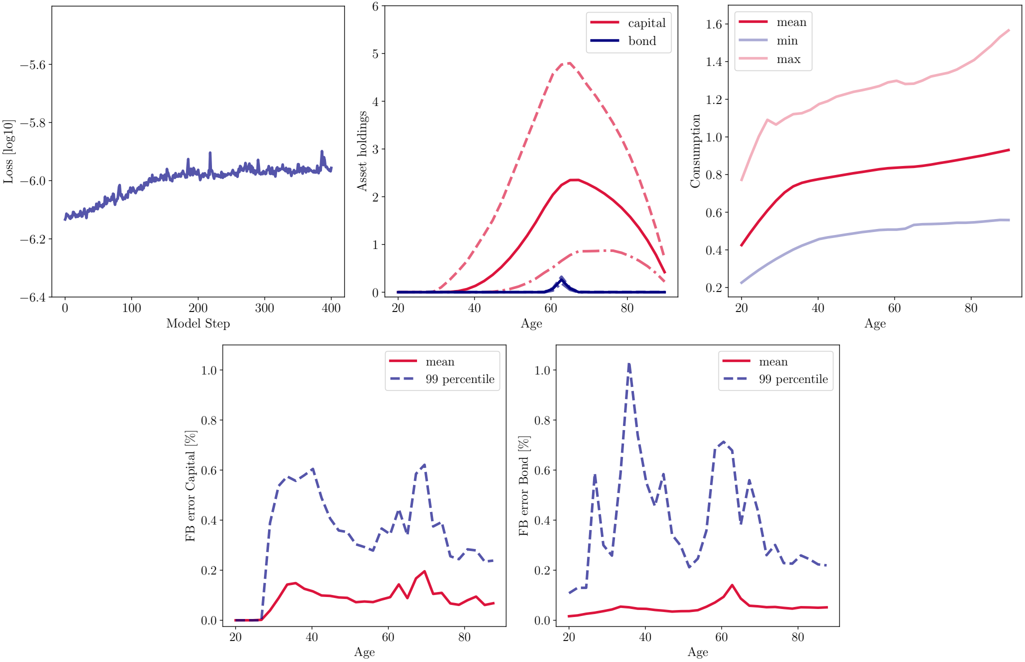
- Loss function

$$\ell_{\boldsymbol{\rho}}(\mathbf{x}_t) := 1 \times \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{k,h} \right)^2 \right)}_{\text{opt. cond. cap.}} + \underbrace{1 \times}_{\text{bond equ. cond. active}} \underbrace{\left(\sum_{h=1}^{H-1} \left(\epsilon_t^{b,h} \right)^2 \right)}_{\text{opt. cond. bond}}$$

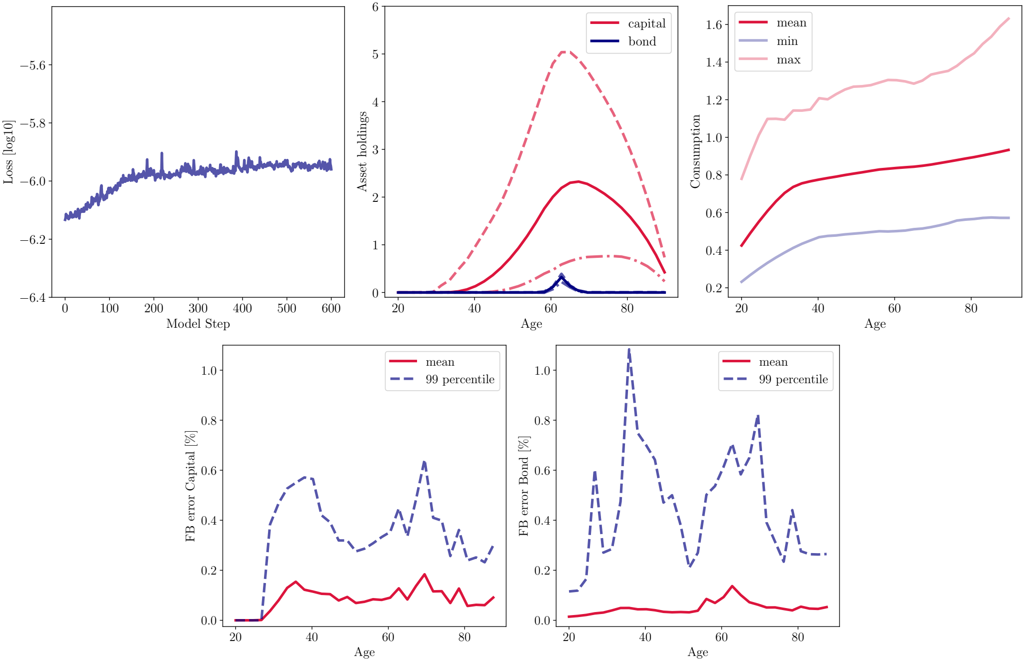


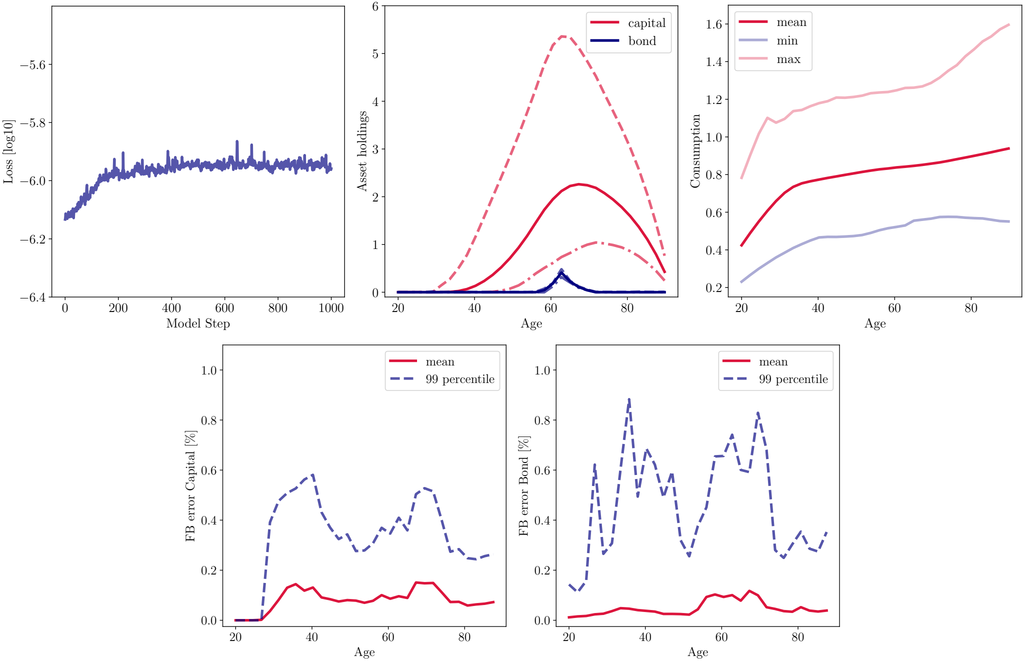
Loss [log10]
Model Step
Asset holdings
Age
capital
bond
Consumption
mean
min
max
FB error Capital [%]
99 percentile
FB error Bond [%]

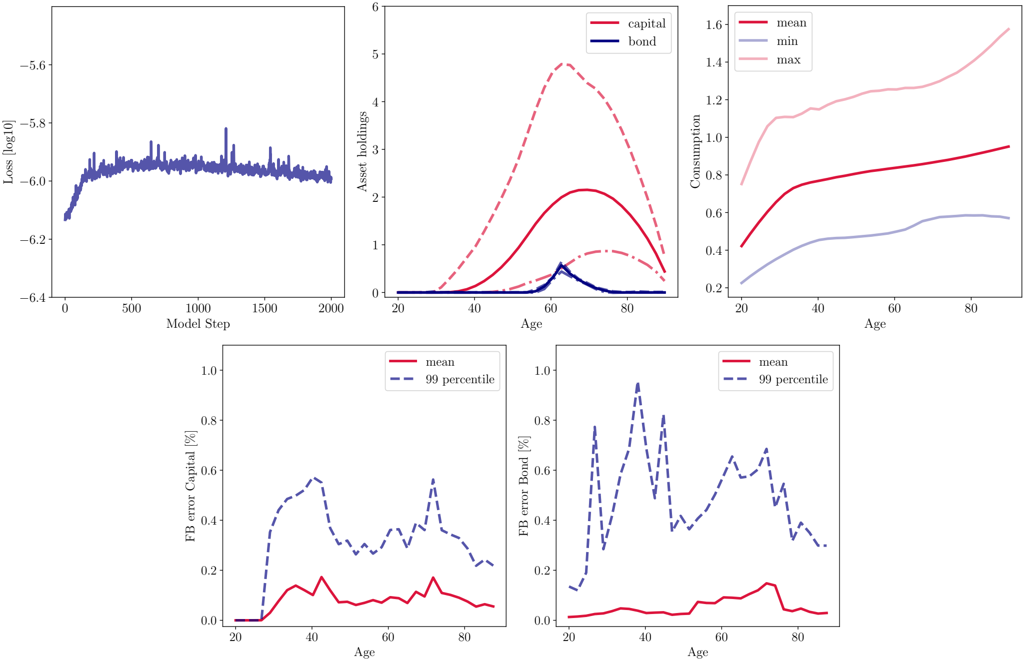

Loss [log10]
Model Step
capital
bond
Asset holdings
Age
mean
min
max
Consumption
Age
mean
99 percentile
FB error Capital [%]
Age
mean
99 percentile
FB error Bond [%]
Age



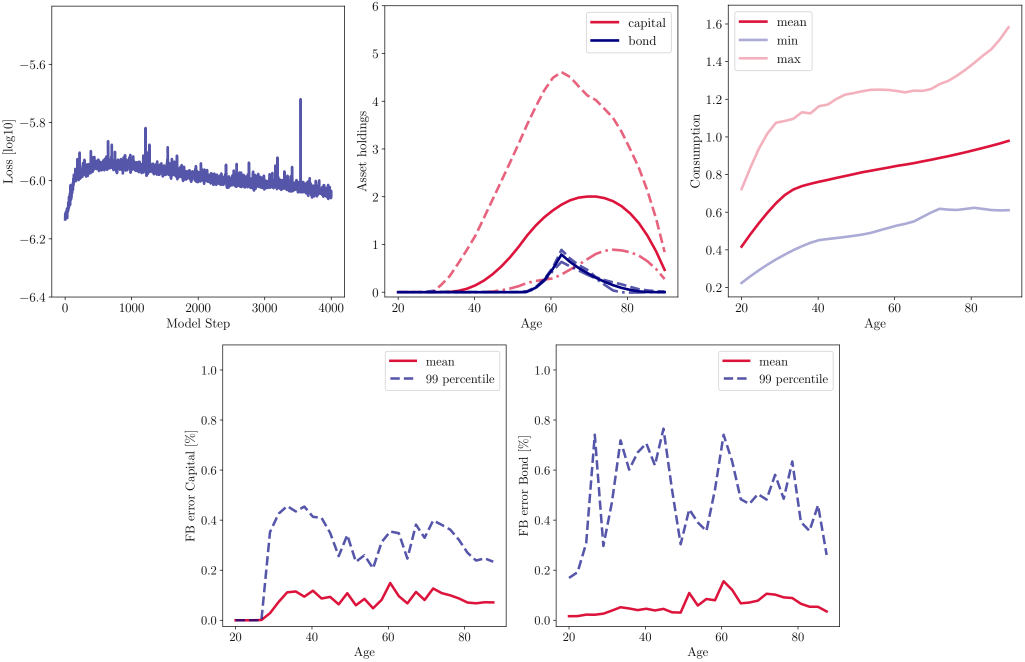
Loss [log10]
Model Step
Asset holdings
Age
capital
bond
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]
mean
99 percentile

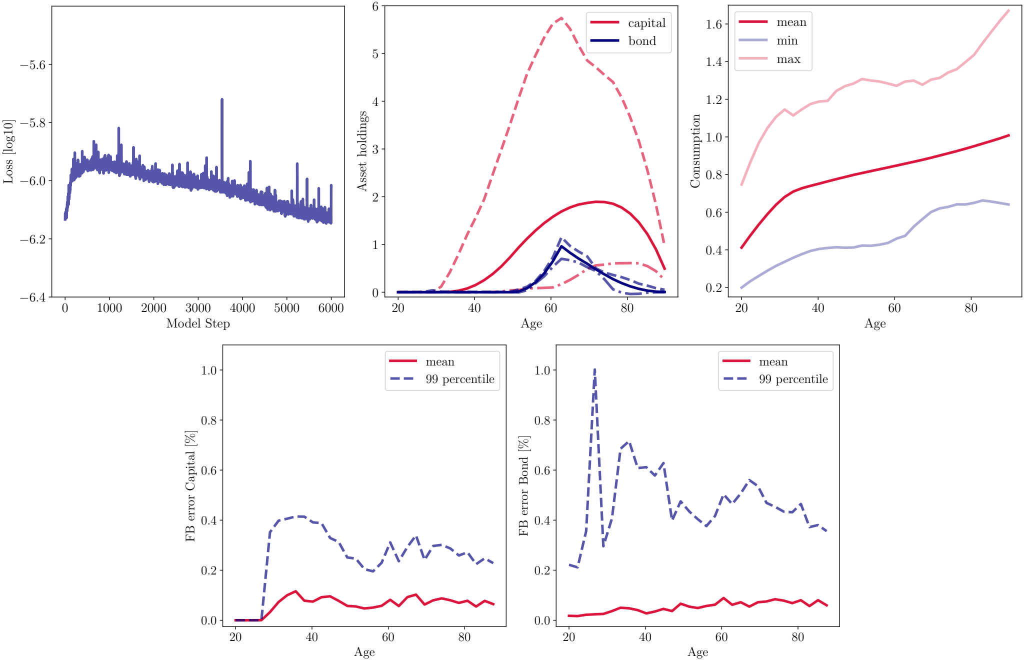

Loss [log10]
Model Step
Asset holdings
Age
capital
bond
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]
mean
99 percentile

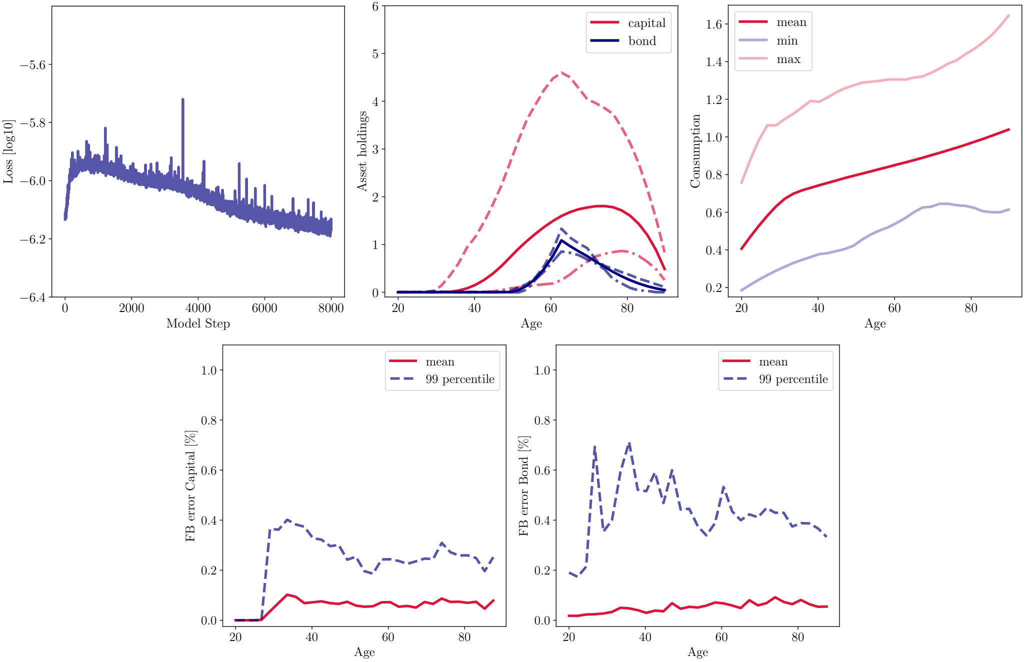

Loss [log10]
Model Step
−5.6
−5.8
−6.0
−6.2
−6.4
0
200
400
600
800
1000
Asset holdings
Age
capital
bond
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]
mean
99 percentile

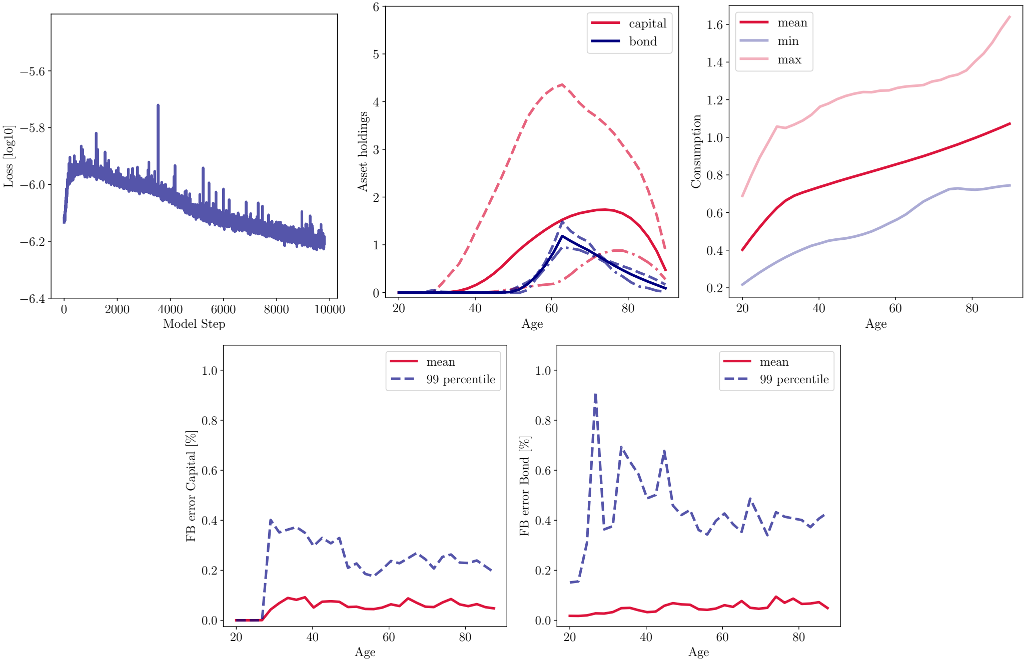


Loss [log10]
Model Step
capital
bond
Asset holdings
Age
mean
min
max
Consumption
Age
mean
99 percentile
FB error Capital [%]
Age
mean
99 percentile
FB error Bond [%]
Age


Loss [log10]
Model Step
Asset holdings
Age
capital
bond
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]


Loss [log10]
Model Step
capital
bond
Asset holdings
Age
mean
min
max
Consumption
Age
mean
99 percentile
FB error Capital [%]
Age
mean
99 percentile
FB error Bond [%]
Age


Loss [log10]
−5.6
−5.8
−6.0
−6.2
−6.4
0
2000
4000
6000
8000
Model Step
Asset holdings
capital
bond
Age
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]


Loss [log10]
Model Step
Asset holdings
capital
bond
Age
Consumption
mean
min
max
FB error Capital [%]
mean
99 percentile
FB error Bond [%]

Step 4: Training with the final supply

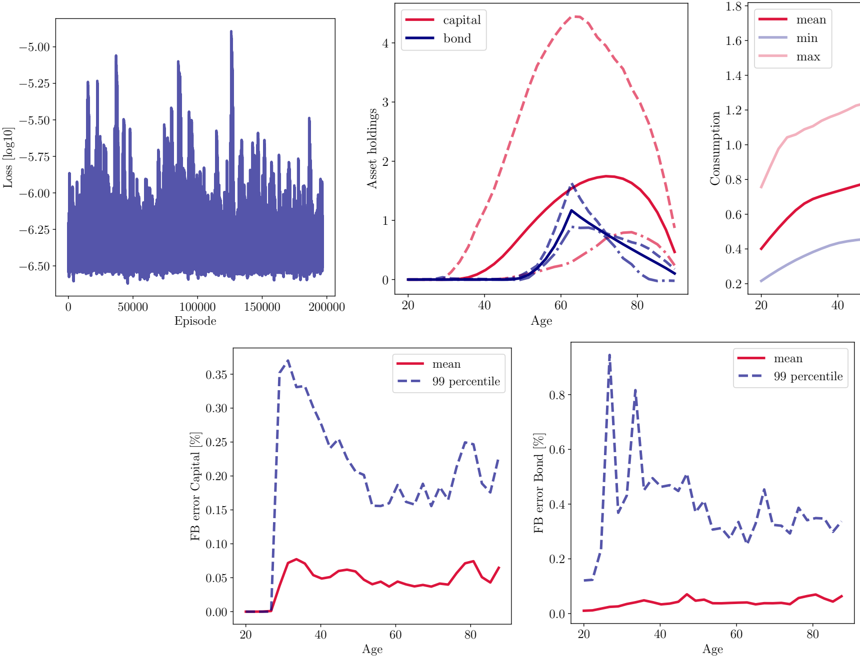
- Borrowing constraint $\underline{b} = 0$, bond at full net-supply from $B = 10$
- Neural network predicts

$$\mathcal{N}_{\rho}^{\text{pre}}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, \underbrace{0.01\times\tilde{b}^1_{t+1}, \ldots, 0.01\times\tilde{b}^{32}_{t+1}}_{\text{bond policies active}}, \hat{p}^b_t]$$

$$\Rightarrow \mathcal{N}_{\rho}(\mathbf{x}_t) = [\hat{k}^1_{t+1}, \ldots, \hat{k}^{32}_{t+1}, \underbrace{\hat{b}^1_{t+1}, \ldots, \hat{b}^{32}_{t+1}}_{\text{always add up the B}}, \hat{p}^b_t]$$

- Loss function contains all remaining equilibrium conditions

$$\ell_{\rho}(\mathbf{x}_t) := 1\times \underbrace{\left(\sum_{h=1}^{H-1}\left(\epsilon_t^{k,h}\right)^2\right)}_{\text{opt. cond. cap.}} + 1\times \underbrace{\left(\sum_{h=1}^{H-1}\left(\epsilon_t^{b,h}\right)^2\right)}_{\text{opt. cond. bond}}$$



Loss [log10]
−5.00
−5.25
−5.50
−5.75
−6.00
−6.25
−6.50
0
50000
100000
150000
200000
Episode
Asset holdings
capital
bond
0
1
2
3
4
20
40
60
80
Age
Consumption
mean
min
max
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
20
40
60
80
Age
FB error Capital [%]
mean
99 percentile
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
20
40
60
80
Age
FB error Bond [%]
mean
99 percentile
0.0
0.2
0.4
0.6
0.8
20
40
60
80
Age

Conclusion

Conclusion

- Deep neural networks are promising to approximate nonlinear functions on high-dimensional domains
- Key ideas in Azinovic et al. (2022):
  - minimizing the error in the equilibrium conditions allows training the neural network without labeled data $\Rightarrow$ neural network can be trained on billions of states
  - training on the simulated path $\Rightarrow$ focus training on where it matters
- Models with many assets remain challenging. To address this issue Azinovic and Žemlička (2023) introduce key innovations
  - market clearing layers, an economics-inspired neutral network architecture
  - step-wise model transformation procedure to guide network training with many assets

Conclusion

- Deep neural networks are promising to approximate nonlinear functions on high-dimensional domains
- Key ideas in Azinovic et al. (2022):
  - minimizing the error in the equilibrium conditions allows training the neural network without labeled data $\Rightarrow$ neural network can be trained on billions of states
  - training on the simulated path $\Rightarrow$ focus training on where it matters
- Models with many assets remain challenging. To address this issue Azinovic and Žemlička (2023) introduce key innovations
  - market clearing layers, an economics-inspired neutral network architecture
  - step-wise model transformation procedure to guide network training with many assets
- Also in the paper: quantitative life-cycle model with disaster risk, housing, equity and bonds in general equilibrium to study the intergenerational consequences of rare disasters (updated version coming soon)

Conclusion

- Deep neural networks are promising to approximate nonlinear functions on high-dimensional domains
- Key ideas in Azinovic et al. (2022):
  - minimizing the error in the equilibrium conditions allows training the neural network without labeled data ⇒ neural network can be trained on billions of states
  - training on the simulated path ⇒ focus training on where it matters
- Models with many assets remain challenging. To address this issue Azinovic and Žemlička (2023) introduce key innovations
  - market clearing layers, an economics-inspired neutral network architecture
  - step-wise model transformation procedure to guide network training with many assets
- Also in the paper: quantitative life-cycle model with disaster risk, housing, equity and bonds in general equilibrium to study the intergenerational consequences of rare disasters (updated version coming soon)
- Other cool papers on deep learning based solution methods: Maliar et al. (2021); Kase et al. (2023); Gu et al. (2023); Kahou et al. (2021); Han et al. (2022); Valaitis and Villa (2024); Kahou et al. (2022); Fernández-Villaverde et al. (2023); Barnett et al. (2023); Jungerman (2023); Kahou et al. (2024)

Thank you!

References I

Azinovic, M., Gaegauf, L., and Scheidegger, S. (2022). Deep equilibrium nets. *International Economic Review*, 63(4):1471–1525.

Azinovic, M. and Žemlička, J. (2023). Economics-inspired neural networks with stabilizing homotopies. *arXiv preprint arXiv:2303.14802*.

Barnett, M., Brock, W., Hansen, L. P., Hu, R., and Huang, J. (2023). A deep learning analysis of climate change, innovation, and uncertainty. *arXiv preprint arXiv:2310.13200*.

Fernández-Villaverde, J., Hurtado, S., and Nuno, G. (2023). Financial frictions and the wealth distribution. *Econometrica*, 91(3):869–901.

Gu, Z., Lauriere, M., Merkel, S., and Payne, J. (2023). Deep learning solutions to master equations for continuous time heterogeneous agent macroeconomic models.

Han, J., Yang, Y., et al. (2022). Deepham: A global solution method for heterogeneous agent models with aggregate shocks. *arXiv preprint arXiv:2112.14377*.

Jungerman, W. (2023). Dynamic monopsony and human capital. Technical report, mimeo.

Kahou, M. E., Fernández-Villaverde, J., Gómez-Cardona, S., Perla, J., and Rosa, J. (2022). Spooky boundaries at a distance: Exploring transversality and stability with deep learning.

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

Kahou, M. E., Yu, J., Perla, J., and Pleiss, G. (2024). How inductive bias in machine learning aligns with optimality in economic dynamics. *arXiv preprint arXiv:2406.01898*.

References II

Kase, H., Melosi, L., and Rottner, M. (2023). Estimating nonlinear heterogeneous agents models with neural networks. *CEPR Discussion Paper No. DP17391*.

Maliar, L., Maliar, S., and Winant, P. (2021). Deep learning for solving dynamic economic models. *Journal of Monetary Economics*, 122:76–101.

Valaitis, V. and Villa, A. (2024). A machine learning projection method for macro-finance models. *Forthcoming in Quantitative Economics*.

Deep Neural Networks

What is a deep neural net?

Consider:

$$\textbf{input} := \mathbf{x} \to W_{\boldsymbol{\rho}}^{1}\mathbf{x} + \mathbf{b}_{\boldsymbol{\rho}}^{1} =: \textbf{hidden 1}$$

What is a deep neural net?

Consider:

$$\begin{aligned} &\textbf{input} := \mathbf{x} \to W^1_{\boldsymbol{\rho}}\mathbf{x} + \mathbf{b}^1_{\boldsymbol{\rho}} =: \textbf{hidden 1} \\ &\to \textbf{hidden 1} \to W^2_{\boldsymbol{\rho}}(\textbf{hidden 1}) + \mathbf{b}^2_{\boldsymbol{\rho}} =: \textbf{hidden 2} \end{aligned}$$

What is a deep neural net?

Consider:

$$\begin{aligned} &\textbf{input} := \mathbf{x} \to W_{\boldsymbol{\rho}}^{1}\mathbf{x} + \mathbf{b}_{\boldsymbol{\rho}}^{1} =: \textbf{hidden 1} \\ \to &\textbf{ hidden 1} \to W_{\boldsymbol{\rho}}^{2}(\textbf{hidden 1}) + \mathbf{b}_{\boldsymbol{\rho}}^{2} =: \textbf{hidden 2} \\ \to &\textbf{ hidden 2} \to W_{\boldsymbol{\rho}}^{3}(\textbf{hidden 2}) + \mathbf{b}_{\boldsymbol{\rho}}^{3} =: \textbf{output} \end{aligned}$$

What is a deep neural net?

Consider:

$$\begin{aligned} &\textbf{input} := \mathbf{x} \to W_{\boldsymbol{\rho}}^1 \mathbf{x} + \mathbf{b}_{\boldsymbol{\rho}}^1 =: \textbf{hidden 1} \\ \to &\textbf{ hidden 1} \to W_{\boldsymbol{\rho}}^2(\textbf{hidden 1}) + \mathbf{b}_{\boldsymbol{\rho}}^2 =: \textbf{hidden 2} \\ \to &\textbf{ hidden 2} \to W_{\boldsymbol{\rho}}^3(\textbf{hidden 2}) + \mathbf{b}_{\boldsymbol{\rho}}^3 =: \textbf{output} \end{aligned}$$

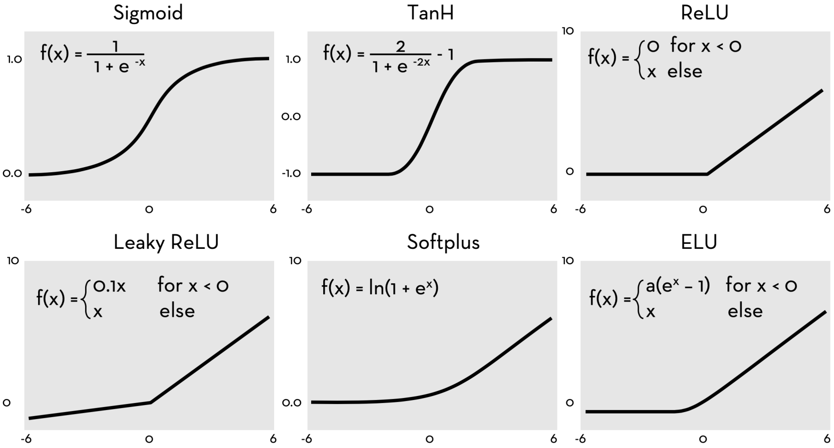
The parameters $\boldsymbol{\rho}$ of this procedure are the entries of the matrices $(W_{\boldsymbol{\rho}}^1,\ W_{\boldsymbol{\rho}}^2,\ W_{\boldsymbol{\rho}}^3)$ and vectors $(\mathbf{b}_{\boldsymbol{\rho}}^1,\ \mathbf{b}_{\boldsymbol{\rho}}^2,\ \mathbf{b}_{\boldsymbol{\rho}}^3)$.

What is a deep neural net? (cont.)

So far we have a concatenation of affine maps and therefore an afffine map.

What is a deep neural net? (cont.)

So far we have a concatenation of affine maps and therefore an afffine map.
Next ingredient: activation functions ϕ^1, ϕ^2, ϕ^3. Activation functions could be any function, but popular are:



What is a deep neural net? (cont.)

Now we get:

$$\begin{aligned} &\textbf{input} := \mathbf{x} \to \phi^1(W_{\boldsymbol{\rho}}^1\mathbf{x} + \mathbf{b}_{\boldsymbol{\rho}}^1) =: \textbf{hidden 1} \\ \to\ &\textbf{hidden 1} \to \phi^2(W_{\boldsymbol{\rho}}^2(\textbf{hidden 1}) + \mathbf{b}_{\boldsymbol{\rho}}^2) =: \textbf{hidden 2} \\ \to\ &\textbf{hidden 2} \to \phi^3(W_{\boldsymbol{\rho}}^3(\textbf{hidden 2}) + \mathbf{b}_{\boldsymbol{\rho}}^3) =: \textbf{output} \end{aligned}$$

The neural net is then given by the choice of activation functions and the parameters $\boldsymbol{\rho}$.

back

Why neural networks?

Approximation method	High-dimensional input	Can resolve local features accurately	Irregularly shaped domain	Large amount of data
Polynomials	✓	✗	✓	✓
Splines	✗	✓	✗	✓
Adaptive (sparse) grids	✓	✓	✗	✓
Gaussian processes	✓	✓	✓	✗
Deep neural networks	✓	✓	✓	✓

Table: Taken from Azinovic et al. (2022).

back

Innovation 1: Details on the market clearing transformation function

- Simple market clearing layer: subtract excess demand ED_t from initial predictions

$$ED_t := \sum_{h \in \mathcal{H}} \tilde{b}^h_{t+1} - B$$

$$\hat{b}^h_{t+1} := \tilde{b}^h_{t+1} - \frac{1}{H} ED_t$$

- Why this adjustment?

$\rightarrow$ we try to minimize the modification to the initial predictions $\{\tilde{b}^h_{t+1}\}_{h \in \mathcal{H}}$.

- Final predictions $\{\hat{b}^h_{t+1}\}_{h \in \mathcal{H}}$ solve

$$\underset{\{x^h_{t+1}\}_{h \in \mathcal{H}}}{\arg\min} \sum_{h \in \mathcal{H}} \left(x^h_{t+1} - \tilde{b}^h_{t+1} \right)^2$$

subject to

$$\sum_{h \in \mathcal{H}} x^h_{t+1} = B$$

Innovation 1: Details on the market clearing transformation function

- Simple market clearing layer: subtract excess demand ED_t from initial predictions

$$ED_t := \sum_{h \in \mathcal{H}} \tilde{b}^h_{t+1} - B$$

$$\hat{b}^h_{t+1} := \tilde{b}^h_{t+1} - \frac{1}{H} ED_t$$

- Why this adjustment?

$\rightarrow$ we try to minimize the modification to the initial predictions $\{\tilde{b}^h_{t+1}\}_{h \in \mathcal{H}}$.

- Final predictions $\{\hat{b}^h_{t+1}\}_{h \in \mathcal{H}}$ solve

$$\underset{\{x^h_{t+1}\}_{h \in \mathcal{H}}}{\arg\min} \sum_{h \in \mathcal{H}} \left(x^h_{t+1} - \tilde{b}^h_{t+1} \right)^2$$

subject to

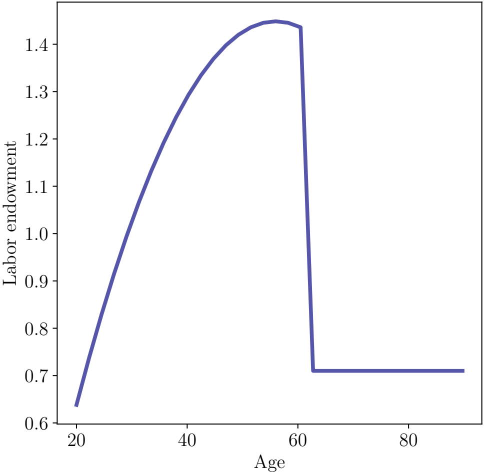
$$\sum_{h \in \mathcal{H}} x^h_{t+1} = B$$

- In the paper: enforcing market clearing & borrowing constraints using implicit layer

back

Parameters

Parameters	H	β	γ	ψ	ρ	σ	α
Values	32	0.912	4	0.1	0.693	0.052	0.333
Meaning	num. age groups	patience	RRA	adj. costs	pers. tfp	std. innov. tfp	cap. share



Households' optimality conditions

$$\left.\begin{aligned} 1 &= \frac{\beta \mathrm{E}\left[u'(c^{h+1}_{t+1})(1-\delta_{t+1}+r_{t+1}+2\psi^k\Delta^{h+1}_{k,t+1}\right]+\mu^h_t}{(1+2\psi^k\Delta^h_{k,t})u'(c^h_t)} \\ k^h_t &\geq 0 \\ \mu^h_t &\geq 0 \\ k^h_t\mu^h_t &= 0 \end{aligned}\right\} \Leftrightarrow \epsilon^{k,h}_t := \psi^{FB}\left(\frac{u'^{-1}\left(\beta \mathrm{E}\left[u'(c^{h+1}_{t+1})\frac{(1-\delta_{t+1}+r_{t+1}+2\psi^k\Delta^{h+1}_{k,t+1}}{(1+2\psi^k\Delta^h_{k,t})}\right]\right)}{c^h_t}-1, \frac{k^h_t}{c^h_t}\right)$$

$$\left.\begin{aligned} 1 &= \frac{\beta \mathrm{E}\left[u'(c^{h+1}_{t+1})\right]+\lambda^h_t}{p^b_t u'(c^h_t)} \\ b^h_t - \underline{b} &\geq 0 \\ \lambda^h_t &\geq 0 \\ (b^h_t - \underline{b}^h)\lambda^h_t &= 0 \end{aligned}\right\} \Leftrightarrow \epsilon^{b,h}_t := \psi^{FB}\left(\frac{u'^{-1}\left(\beta \mathrm{E}\left[\frac{1}{p^b_t}u'(c^{h+1}_{t+1})\right]\right)}{c^h_t}-1, \frac{b^h_t-\underline{b}}{c^h_t}\right)$$

where

$$\psi^{FB}(a, b) := a + b - \sqrt{a^2 + b^2}$$

back