Deep Learning Solutions to Master Equations for Continuous Time Heterogeneous Agent Macroeconomic Models

Zhouzhou Gu

Princeton

joint work with Mathieu Laurière, *Sebastian* Merkel *and Jonathan* Payne

Macro-finance Workshop (USC Marshall)

August 22, 2023

Introduction

- We solve continuous time, heterogeneous agent economies:
  - Consider economies with aggregate shocks, long-term assets, portfolio choice, illiquidity.
  - Consider different finite dimensional approximations to the distribution (finite agents, projection).
  - Solve the resulting high dimensional PDE(s) using neural network approximations.
- *How do we test it?* compare to solutions for canonical economic models.
  (e.g. Aiyagari (1994), Krusell and Smith (1998), Basak and Cuoco (1998), and extensions).
- *What economic question do we answer?* the impact of housing policy on inequality.
  (e.g. in preliminary follow-up paper Gu & Payne (2023) "Housing and Inequality").

Outline

1. Baseline Hetero-Agent Macroeconomic Models

2. Model Solution Approach

3. Conclusion

Outline

Aiyagari-Bewley-Huggett and Krusell-Smith Models

- Goal:
  - Model “Precautionary Saving" into assets:
    (e.g. money, bonds,...)
  - Explore the impact on aggregate economy
    (e.g. capital overaccumulation, monetary/fiscal policy)
- Idea: “saving for a rainy day"
  - Households get idiosyncratic shocks to employment
  - Cannot fully insure against (markets are incomplete)
  - Instead, must build savings to they still have income when unemployed

Environment: Households

- Continuous time, infinite horizon economy.
- Agents are heterogeneous in wealth a, and labor hours y. Solve consumption-saving problem:

$$\max_{\{c_t\}_{t\geq 0}} \mathbb{E} \int_0^\infty e^{-\rho t} u(c_t) dt \; s.t.$$

$$\begin{aligned} da_t &= (w_t y_t + r_t a_t - c_t)dt \\ y_t &\in \{y_1, y_2\} \\ a_t &\geq \underline{a} \end{aligned} \tag{1}$$

  - c_t: consumption
  - u: utility function, $u' > 0, u'' < 0$
  - ρ: discount rate
  - r_t: capital return, w_t: wage rate
  - y_t: hours workd, switching rate $\{\lambda_1, \lambda_2\}$
  - a_t: wealth
  - $\underline{a}$: borrowing limit.

Environment: Firms

- Representative firm with production technology. Solves:

$$\max_{K_t, L_t} \{e^{Z_t} K_t^{\alpha} L_t^{1-\alpha} - (r_t + \delta) K_t - w_t L_t\} \tag{2}$$

  - Z_t: productivity, follows exogenous mean reverting aggregate process:
    (Aside: Aiyagari-Bewley-Huggett: $Z_t = \bar{Z}$ is fixed; Krusell-Smith (1998): Z_t is stochastic.)

$$dZ_t = \eta(\overline{Z} - Z_t)dt + \sigma^z dB_t^0$$

  - K_t: capital (rented from competitive capital market at r_t)
  - L_t: labor (rented from competitive labor market at r_t)

- Optimality conditions:

$$r_t = \alpha e^{Z_t}(L_t/K_t)^{1-\alpha}, \quad w_t = (1-\alpha)e^{Z_t}(K_t/L_t)^{\alpha}$$

Environment: Markets

- Capital market clearing:
$$K_t = \sum_{i=1}^{2} \int_{\underline{a}}^{\infty} a g_t(a, y_i) da$$
- Labor market clearing:
$$L_t = \sum_{i=1}^{2} \int_{\underline{a}}^{\infty} y_i g_t(a, y_i) da$$
- Question: *Why does the distribution matter? Can we only consider first moments?*
Answer: need law of motion for population distributions to understand moment dynamics.

Equilibrium

Definition: Given an initial density g_0, an equilibrium for this economy consists of a collection of stochastic process $\{c_t^i, g_t, r_t, w_t, z_t : t \geq 0, i \in I\}$, such that:

1. Given their beliefs about the price process $(\hat{r}, \hat{w}) = \{\hat{r}_t, \hat{w}_t : t \geq 0\}$:
   - each agent's control process, c_t^i, solves problem (1),
   - representative firm solves problem (2).
2. The prices r_t, w_t satisfy market clearing condition, and belief consistency: $r_t = \hat{r}_t, w_t = \hat{w}_t$.

Master Equation: Recursive Representation

- Aggregate states: $\{z, g\}$, individual states: $\{a, y\}$, household value function: $V(a, y, z, g)$.
- Household choose c to solve Hamilton-Jacobi-Bellman Equation:

$$
\begin{aligned}
0 = \max_{c \geq 0} \Big\{ & -\rho V(a,y,z,g) + u(c) + \partial_a V(a,y,z,g) s(a,y,c,r,w) \\
& + \lambda(y)\left(V(a,\tilde{y},z,g) - V(a,y,z,g)\right) + \partial_z V(a,y,z,g)\mu^z(z) + 0.5\left(\sigma^z\right)^2 \partial_{zz} V(a,y,z,g)\Big\} \\
& + \int_{\mathcal{X}} \hat{\mu}_g(\hat{c}_t, z_t, g_t) \frac{\partial V}{\partial g}(a,y,z,g)(y)dy \quad s.t. \quad \text{BC: } \frac{\partial V}{\partial a}\Big|_{a=\underline{a}} \geq u'(wy + r\underline{a})
\end{aligned}
$$

- For optimal policy rule, $c^*(x, z, g; \hat{\mu})$, for z_t, population density, g, law of motion:

$$
dg_t(x) = \underbrace{\mu_g(c^*(a_t, y_t, z_t, g_t), z_t, g_t)dt}_{\text{Operator } (\mathcal{L}^k g)(x,z,g)}. \tag{3}
$$

Master Equation: Define Operators

- Master Equation: one equation includes everything: recursive representation and imposed equilibrium (*including prices and belief consistency*).
- Replace Hard constraint $a \geq \underline{a}$ by Penalty function: $\mathbf{1}_{a \leq a} \psi(a), \psi(\cdot) < 0$
- Household's Master Equation

$$0 = \max_{c \geq 0} \Big\{ -\rho V(a,y,z,g) + u(c) + \mathbf{1}_{a \leq \underline{a}} \psi(a) + \partial_a V(a,y,z,g) s(a,y,c,r,w)$$

$$\underbrace{+\lambda(y)\left(V(a,\tilde{y},z,g) - V(a,y,z,g)\right) + \partial_z V(a,y,z,g)\mu^z(z) + 0.5\left(\sigma^z\right)^2 \partial_{zz} V(a,y,z,g)}_{=:\text{ Operator } (\mathcal{L}^h V)(x,z,g)} \Big\}$$

$$+ \underbrace{\int_{\mathcal{X}} \mu_g(\hat{c}_t, z_t, g_t) \frac{\partial V}{\partial g}(a,y,z,g)(y) dy}_{=:\text{ Operator } (\mathcal{L}^g V)(x,z,g)}$$

"Master Equation" and Operators

- To summarize notation, we want to solve $0 = (\mathcal{L}V)(a, y, z, g)$ where:
  - $\mathcal{L} = \mathcal{L}^h + \mathcal{L}^g$ is the operator for total Master equation
  - $\mathcal{L}^h$ is the "standard" HJBE operator capturing how households optimize
  - $\mathcal{L}^g$ captures how the distribution impacts household value (the "hard" part!)
  - $\mathcal{L}^k$ is the operator for how distribution evolves.

Outline

1. Baseline Hetero-Agent Macroeconomic Models

2. Model Solution Approach
   - Distribution Approximations
   - Neural Network Approximation
   - Training Algorithm
   - Numerical Results

3. Conclusion

Outline

- Goal: "solve Master equation numerically"
- Problem: Master equation contains an infinite dimensional derivative in $\mathcal{L}^g$.
- Solution: three main ingredients:
    1. High but finite dimensional approximation to distribution and Master equation (finite population or projection),
    2. Parameterize V by neural network, and
    3. Train the parameters to minimize the (approximate) master equation residual.

Outline

1. Baseline Hetero-Agent Macroeconomic Models

2. Model Solution Approach
   - Distribution Approximations
   - Neural Network Approximation
   - Training Algorithm
   - Numerical Results

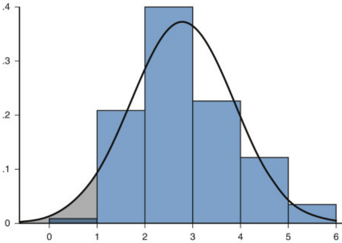
3. Conclusion

Approach A: Finite Population

- Replace distribution g_t by finite number of price taking agents $\hat{g}_t := \{x_t^i : i \leq I\}$.
- Agent $i \leq I$ behaves as if their individual actions do not influence prices.
- Law of motion for distribution becomes evolution of other agent j's states.

Approach B: Projection Onto Basis

- Approximate the distribution $g_t(x)$ by $\sum_{i=1}^{N} \alpha_t^i h^i(x)$, where:
  - α_t^i is a time varying coefficient, $h^i(x)$ is basis function, and
  - Example bases: Indicator Functions, Chebyshev polynomial, Eigenfunctions, ...
  - Distribution characterized by coefficients: $\hat{g}_t := \{\alpha_t^1, ...\alpha_t^N\}$.
- We approximate the distribution by a histogram:
  - Basis is a collection of N^x points: $x_1, \ldots, x_{N^x}$, in $\mathcal{X}$.
  - We approximate g_t by a vector $\alpha_t \in \mathbb{R}^{N^x}$ of mass points at $x_1, \ldots, x_{N^x}$.
- Law of motion of the mass points is the finite difference approximation of (3).



Related Literature

Traditional methods for heterogeneous agent models with aggregate shocks:

- Fit a statistical approximation to the law of motion for the key aggregate state variables (e.g. [Krusell and Smith, 1998], [Den Haan, 1997], [Fernández-Villaverde et al., 2023])
- Linear perturbation in the aggregate state and then solve the resulting linear problem with matrix algebra (e.g. [Reiter, 2002], [Reiter, 2008], [Reiter, 2010], [Winberry, 2018], [Ahn et al., 2018], [Bhandari et al., 2023])
- Low dimensional projection of the distribution (e.g. [Reiter, 2009], [Prohl, 2017], [Schaab, 2020])
- *This paper:* high dimensional, global solution.

Machine learning for macro-economic models and MFGs:

- Discrete time (e.g. [Azinovic et al., 2022], [Han et al., 2021], [Maliar et al., 2021], [Kahou et al., 2021], [Bretscher et al., 2022], [Fernandez-Villaverde et al., 2020], [Wagner, 2023])
- Continuous time (e.g. [Duarte, 2018], [Gopalakrishna, 2021])
- Controls or value functions in MFGs (e.g. [Perrin et al., 2022, Germain et al., 2022, Laurière, 2021])
- *This paper:* continuous time model with distributions.

Outline

Neural Network

- A neural network is a type of parametric functional approximation that is built by composing affine and non-linear functions in a chain or "network" (see [Goodfellow et al., 2016])
- Let $\hat{X} := \{x, z, \hat{g}\}, x \equiv \{a, y\}$ denote the collection of inputs into the approximate value function.
- Denote neural network approximation to the value function by $V(\hat{X}) \approx \hat{V}(\hat{X}; \theta)$,
  - Where θ are the parameters in the neural network approximation,
  - Ultimately, we will "learn" the parameters, θ, that give "best" approximation
- Many types of neural networks:
  - Our "default" is a "fully connected" "feedforward" network,

Feedforward Neural Network

$$\begin{aligned}
\boldsymbol{h}^{(1)} &= \phi^{(1)}(W^{(1)}\hat{\boldsymbol{X}} + \boldsymbol{b}^{(1)}) && \ldots \text{Hidden layer 1} \\
\boldsymbol{h}^{(2)} &= \phi^{(2)}(W^{(2)}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}) && \ldots \text{Hidden layer 2} \\
&\vdots \\
\boldsymbol{h}^{(H)} &= \phi^{(H)}(W^{(H)}\boldsymbol{h}^{(H-1)} + \boldsymbol{b}^{(H)}) && \ldots \text{Hidden layer H} \\
\boldsymbol{o} &= W^{(H+1)}\boldsymbol{h}^{(H)} + \boldsymbol{b}^{(H+1)} && \ldots \text{Output layer} \\
\hat{V} &= \phi^{H+1}(\boldsymbol{o}) && \ldots \text{Output}
\end{aligned}$$

- H: is the number of *hidden layers*
- Length of vector $\boldsymbol{h}^{(i)}$: number of *neurons* in hidden layer i
- $\phi^{(i)}$: is the *activation function* for hidden layer i
- σ: is the *activation function* for the output layer
- $\theta = (W^1, \ldots W^{(H+1)}, b^{(1)}, \ldots, b^{(H+1)})$ are the *parameters*

Outline

Generic Algorithm

Starting with an initial θ^0. At iteration n with guess θ^n:

1. Sample $S^n := \{S^{ne}, S^{nb}\}$ from the state space, where:
   - $S^{ne} = \{(x_m, z_m, \hat{g}_m)\}_{m \leq M^e}$ are sample points on interior $x_m \in \mathcal{X}$,
   - S^{nb} are sample points on boundary $x_m \in \partial\mathcal{X}$.

2. Calculate the weighted average error:

$$\mathcal{E}(\theta^n, S^n) = \kappa^e \mathcal{E}^e(\theta^n, S^{ne}) + \kappa^b \mathcal{E}^b(\theta^n, S^{nb}) + \kappa^f \mathcal{E}^f(\theta^n, S^{ne}), \quad \text{where}$$

   - $\mathcal{E}^e(\theta^n, S^{ne}) := \frac{1}{M^e} \sum_{m \leq M^e} |\hat{\mathcal{L}}(x_m, z_m, \hat{g}_m)|^2$ is error in Master equation
   - $\mathcal{E}^b(\theta^n, S^{nb})$ is error on boundary (if applicable)
   - $\mathcal{E}^f(\theta^n, S^{ne})$ is penalty for "wrong" shape (e.g. penalty for non-concavity of V)

3. Update the NN parameters using "stochastic" gradient descent (built-in packages in Python):

$$\theta^{n+1} = \theta^n - \alpha_n D_\theta \mathcal{E}(\theta^n, S^n)$$

Neural Network Q & A

- ***Q.*** *Why do we draw new samples each epoch?*
  - Avoid overfitting on random samples.
- ***Q.*** *Could we use an alternative parametric approximation like Chebyshev polynomials?*
  - Automatic derivatives can easily calculated for neural networks.
  - Effective non-linear optimizers have been developed for neural nets.
  - We also find that Chebyshev projections struggle to capture high curvature
- ***Q.*** *Does the algorithm solve for global minimum?*
  - Loss calculated by randomly collected points so we move around the paramater space.
  - Helps to escape from local minimum trap by SGD (but does not seem to prevent it).

Neural Network Q & A

- ***Q.*** *Why do we need shape constraints?*
  - Neural network can find “bad” approximate solutions,
  - Especially likely to find solutions with zero derivative when limited curvature in the problem,
  - Forcing a shape constraint can prevent this.
- ***Q.*** *What about slowing down the updating?*
  - For projection methods, we use “Howard improvement algorithm” to slow down the rate of updating (fix policy rule for some iterations and just update V).
  - [Duarte, 2018] and [Gopalakrishna, 2021] suggest introducing a “false” time step but so far we have not found this necessary (or found a way to implement at high scale).
    - We use shape constraints as a replacement.

Neural Network Q & A

- ***Q.*** *What about imposing symmetry and/or dimension reduction?*
  - [Han et al., 2021] and [Kahou et al., 2021] suggest feeding the distribution through a preliminary neural network that reduces the dimension and imposes symmetry.
  - We find we can solve the problem with and without this approach

Great in Principle But Implementation is Tricky

- Algorithm has some large advantages for continuous time:
  1. Can deal with high dimensional differential equations,
  2. Calculates derivatives using automatic differentiation (rather than finite difference),
  3. Can sample on random points rather instead of grid.
- However, the algorithm is tricky to implement (despite being easy describe)! Implementation details in paper.

Implementation Details

Algorithm 1 Pseudo code

Initialize V, and Optimizer.
while Loss > tolerance **do**
  Sample input: $(X = x_1, x_2..., x_n, z)$
  Construct: $c_i(X)$, equilibrium from i's perspective and then $\mathcal{L}_i^h$
  Diffeq $= \mathcal{L}^h V(X)$
  for $j = 2$ to N **do**
    $X_j = (x_j, x_2, .., x_{n-1}, x_1, x_{n+1}, .., z)$
    Construct: $c_j(X_j)$, equilibrium from j's perspective and then $\mathcal{L}_j^g$
    Diffeq $+ = \mathcal{L}_j^g V(X)$
  end for
  Loss = |Diffeq|
  Optimizer.step()
end while

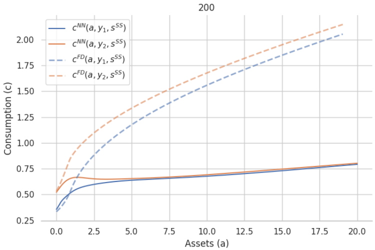
Implementation Details (Pytorch): Automatic Differentiation

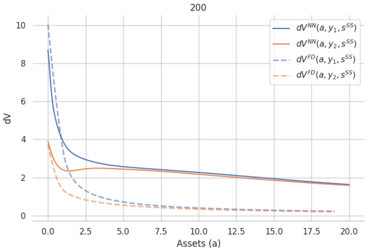
```
def get_derivs_1order(self, y_pred, x):
    """ Uses automatic differentiation to take fall derivatives.
    """
    dy_dx = torch.autograd.grad(y_pred, x, create_graph=True,
        grad_outputs=torch.ones_like(y_pred))[0]
    return dy_dx # Return 'automatic' gradient.

# Example
a.requires_grad_(True)  # Start tracking
Va_pred = model_a(a)
dV_da   = self.get_derivs_1order(Va_pred, a)
a.requires_grad_(False) # Stop tracking
```

Outline

Numerical Results: Policy Plots

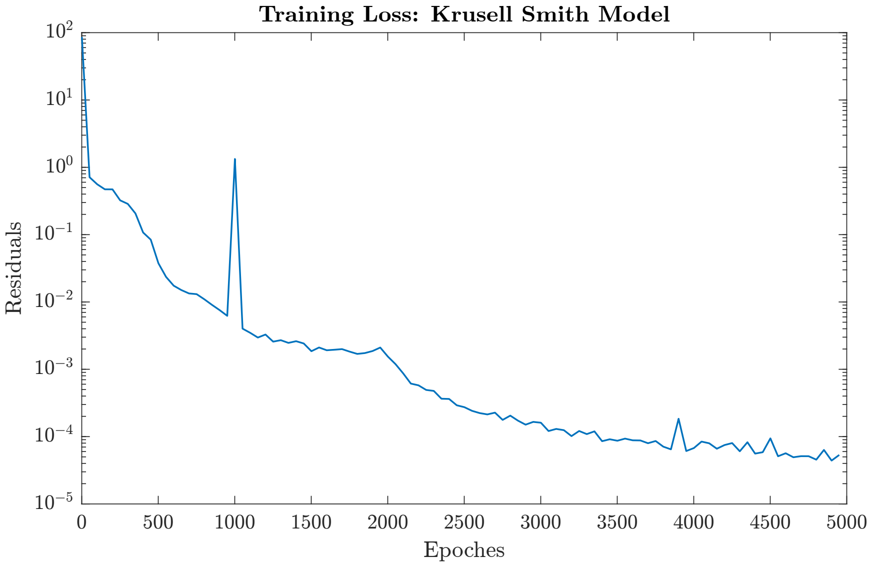
Training of the neural network (FA approach):



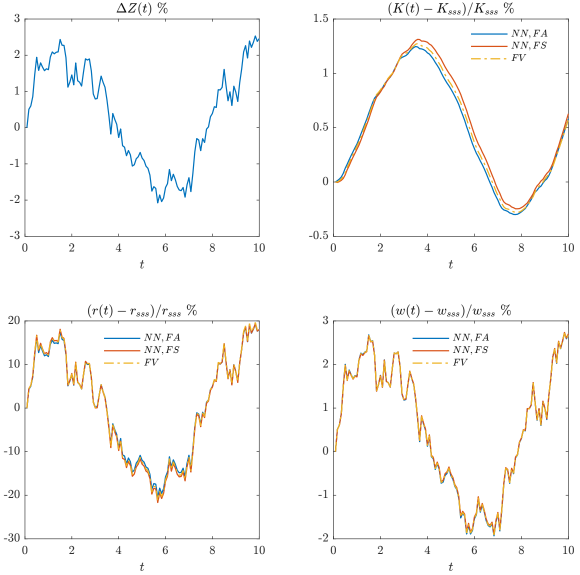


Play/Pause

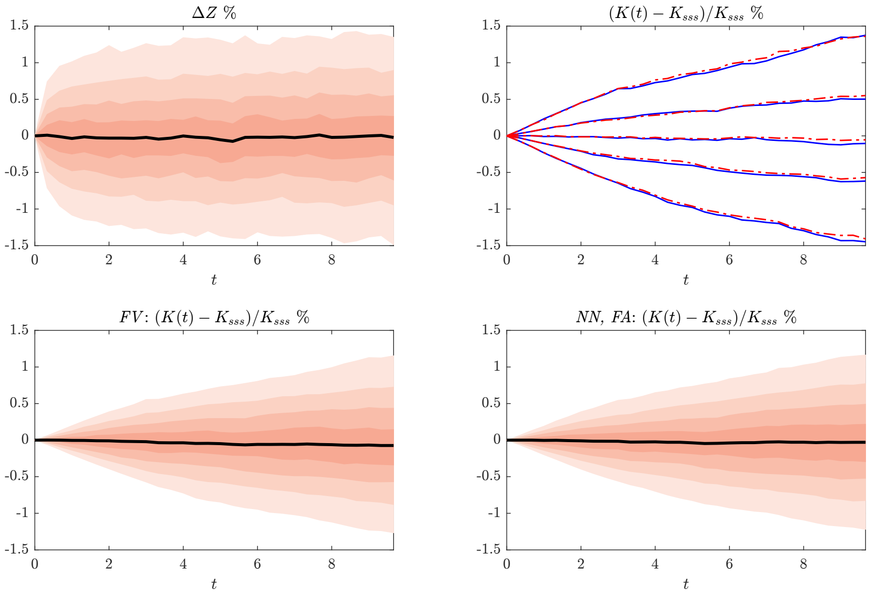
Numerical Results: Losses



Numerical Results: Time-series Plots



Numerical Results: Fancharts



Outline

Practical Lessons

1. Working out the correct sampling approach is very important.
2. Neural networks have difficulty dealing with inequality constraints.
3. Enforcing shape constraints is very important.
4. Mean squared errors can be misleading.
5. Start with a simple model to tune hyperparameters.

Summary and Perspectives

- Master equation approach to tackle aggregate shocks
- Generic algorithm
- Different approaches (Market Clearing condition: simple vs complex)
- Flexibility of the method (Extension to Long-term Assets in Gu and Payne (2023))

References I

[Ahn et al., 2018] Ahn, S., Kaplan, G., Moll, B., Winberry, T., and Wolf, C. (2018).
When inequality matters for macro and macro matters for inequality.
NBER macroeconomics annual, 32(1):1–75.

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

[Bhandari et al., 2023] Bhandari, A., Bourany, T., Evans, D., and Golosov, M. (2023).
A perturbational approach for approximating heterogeneous-agent models.

[Bretscher et al., 2022] Bretscher, L., Fernández-Villaverde, J., and Scheidegger, S. (2022).
Ricardian business cycles.
Available at SSRN.

[Den Haan, 1997] Den Haan, W. (1997).
Solving Dynamic Models with Aggregrate Shocks and Heterogeneous Agents.
Macroeconomic Dynamics, 1(2):355–386.

[Duarte, 2018] Duarte, V. (2018).
Machine learning for continuous-time economics.
Available at SSRN 3012602.

References II

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

[Fernandez-Villaverde et al., 2020] Fernandez-Villaverde, J., Nuno, G., Sorg-Langhans, G., and Vogler, M. (2020).
Solving high-dimensional dynamic programming problems using deep learning.
Unpublished working paper.

[Germain et al., 2022] Germain, M., Laurière, M., Pham, H., and Warin, X. (2022).
DeepSets and their derivative networks for solving symmetric PDEs.
Journal of Scientific Computing, 91(2):63.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

[Gopalakrishna, 2021] Gopalakrishna, G. (2021).
Aliens and continuous time economies.
Swiss Finance Institute Research Paper, (21-34).

[Han et al., 2021] Han, J., Yang, Y., and E, W. (2021).
DeepHAM: A global solution method for heterogeneous agent models with aggregate shocks.
arXiv preprint arXiv:2112.14377.

References III

[Kahou et al., 2021] Kahou, M. E., Fernández-Villaverde, J., Perla, J., and Sood, A. (2021).
Exploiting symmetry in high-dimensional dynamic programming.
Technical report, National Bureau of Economic Research.

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

[Laurière, 2021] Laurière, M. (2021).
Numerical methods for mean field games and mean field type control.
Mean Field Games, 78:221.

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

[Perrin et al., 2022] Perrin, S., Laurière, M., Pérolat, J., Élie, R., Geist, M., and Pietquin, O. (2022).
Generalization in mean field games by learning master policies.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9413–9421.

[Prohl, 2017] Prohl, E. (2017).
Discetizing the Infinite-Dimensional Space of Distributions to Approximate Markov Equilibria with Ex-Post Heterogeneity and Aggregate Risk.

References IV

[Reiter, 2002] Reiter, M. (2002).
Recursive computation of heterogeneous agent models.
manuscript, Universitat Pompeu Fabra, (July):25–27.

[Reiter, 2008] Reiter, M. (2008).
Solving heterogeneous-agent models by projection and perturbation.
Journal of Economic Dynamics and Control, 33:649–665 Contents.

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

[Reiter, 2010] Reiter, M. (2010).
Approximate and Almost-Exact Aggregation in Dynamic Stochastic Heterogeneous-Agent Models.
Technical report, Vienna Institute for Advanced Studies.

[Schaab, 2020] Schaab, A. (2020).
Micro and macro uncertainty.
Available at SSRN 4099000.

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

Outline

4. Long-Term Illiquid Assets and Portfolio Choice (Gu-Payne (2023)

Environment

- Continuous time, infinite horizon economy.
- Consumption good produced by a "Lucas tree" according to stochastic process:

$$dy_t = \eta(\bar{y} - y_t)dt + \sigma dB_t^0, \tag{4}$$

- Assets: bonds in zero net supply, equity in Lucas tree, housing:
  - "Liquid" competitive markets for goods, bonds (at price w_t), and equity (at price q_t).
  - "Illiquid" housing; trading housing at rate $\iota_{i,t}$ incurs transaction cost: $\Psi(\iota_{i,t}, h_{i,t}) = \frac{1}{2}\psi\iota_{i,t}^2/h_{i,t}$
- Population approximated by I of agents (start with finite agent approximation):
  - Get flow utility $u(c_t^i)$ from consuming c_t^i goods and $\zeta_{i,t}\nu(h_{i,t}, a_{i,t})$ from housing $h_{i,t}$, where
  - $\zeta_{i,t} \in \{n_1, n_2\}$ is idiosyncratic housing need ("life-stage") , which switches at rate $\lambda(\zeta_t^i)$.
  - Face collateral borrowing constraint: $a_t \geq -\kappa p_t h_{i,t}$

Agent Problem

- Idiosyncratic states: $x_t^i = [a_t^i, h_t^i, \zeta_t^i]$, a_t^i is liquid wealth, h_t^i is housing, ζ_t^i is housing need.
- Given their beliefs, agent i chooses (c_i, b_i, ι_i) to maximise utility s.t. state evolution:

$$V(x_0^i, z_0) = \max_{c^i, b^i, \iota^i} \mathbb{E}_0 \left[\int_0^\infty e^{-\rho t} (u(c_t^i) + \zeta_{i,t} \nu(h_{i,t}, a_{i,t}) + \mathbf{1}_{a_t \leq -\kappa p_t h} \phi(a_t, h_t)) dt \right], \quad (5)$$

$$s.t. \quad dz_t = \dots, \quad dx_t^i = \dots, \quad \Psi(a,h) := -0.5\psi(a + \kappa p h)^2$$

Recursive Rep by Finite Agent Approach

- Now, the FOCs are given by the following respectively:

$$[c_i]: \quad c_i = (u')^{-1} \left(\frac{\partial V_i}{\partial a_i} \right)$$

$$\begin{aligned} [b_i]: \quad 0 = & -\frac{\partial V}{\partial a_i} \left(r(\cdot) - \mu_q(\cdot) - \frac{y}{q(\cdot)} \right) + \frac{\partial^2 V}{\partial a_i^2} (a_i - b) \sigma_q^2(\cdot) && \text{[Myopic Demand]} \\ & + \frac{\partial^2 V}{\partial a_i \partial y} \sigma_q(\cdot) \sigma_y(\cdot) + \sum_j \frac{\partial^2 V}{\partial a_i \partial a_j} \sigma_q(\cdot) \hat{\sigma}_{a_j}(\cdot) && \text{[Hedging Demand]} \end{aligned} \quad (6)$$

$$[\iota_i]: \quad \iota_i = \frac{h_i}{\psi} \left(\frac{\partial V_i / \partial h}{\partial V_i / \partial a} - p \right)$$

Master Equation Formulation

- Define $\xi_a := \partial_a V_i(x, z, g)$ and $\xi_h := \partial_h V_i(x, z, g)$. After we substitute the equilibrium KFE into the agent optimization, we are left with the following master equations:

$$\rho = \frac{y}{q} + \mu_q + \mu_{\xi_a} + \mathbb{E}[j_{\xi_a}] + \sigma_{\xi_a}\sigma_q + \frac{1}{\xi_a}\frac{\partial \phi}{\partial a} \tag{7}$$

$$\rho = \frac{\partial \Psi}{\partial h} + \mu_{\xi_h} + \mathbb{E}[j_{\xi_h}] + \frac{1}{\xi_h}\frac{\partial \phi}{\partial h} \tag{8}$$

- Where impose asset market clearing to can get r_t and p_t in closed form in terms of ξ_a and ξ_h and z, g (Derivation Let $\mathbf{1} := (\frac{1}{N}, ..., \frac{1}{N})$, $\mathbf{M}_{ij} := \sigma_q^2 \xi_{i,a_j}$):

$$r - r^q = \frac{q + \mathbf{1} \cdot (\mathbf{M}^{-1}\boldsymbol{\xi}_y)\sigma_q\sigma_y}{\mathbf{1} \cdot (\mathbf{M}^{-1}\boldsymbol{\xi})}, \qquad p = \frac{1}{H}\left(\sum_i \frac{\xi_{h,i}}{\xi_{a,i}} h_i\right), \tag{9}$$

- But we only have $q_t = q(z, g)$ implicitly; so only know it must satisfy Ito's lemma. (non-trivial market clearing condition implies a PDE for q)

Algorithm: Losses' Construction

1. **Block 1: Distribution evolution**: Calculate law of motion for the wealth and housing shares $\{\mu_{\eta_i}, \sigma_{\eta_i}, \mu_{\varphi_i}\}_i$.
2. **Block 2: Agent optimization**: Evaluate the weighted L^1 average error:
$$\mathcal{E}^{\xi}(\theta_{\xi}^n, S^{ne}) = \frac{w^a}{M}\sum_{m\leq M}|\mathcal{L}^{hm}/\xi_h^m| + \frac{w^h}{M}\sum_{m\leq M}|\mathcal{L}^{am}/\xi_a^m|. \tag{10}$$
where $\mathcal{L}^{am}$ and $\mathcal{L}^{hm}$ are the error in sample m for the pdes (7) and (8) respectively (the pdes for ξ_a and ξ_h). Update parameters θ_{ξ}^n using stochastic gradient descent.
3. **Block 3: Equilibrium consistency**: Evaluate the weighted average error by goods market clearing condition and consistency:
$$\mathcal{E}^{q}(\theta_{\xi}^n, S^{ne}) = \frac{1}{M}\left(\sum_{m\leq M}\epsilon_c|\sum_i c_i - y| + \epsilon_\mu|\mathcal{L}^{\mu m}| + \epsilon_\sigma|\mathcal{L}^{\sigma m}|\right). \tag{11}$$
where $\mathcal{L}^{\mu m}$ and $\mathcal{L}^{\sigma m}$ are the errors in sample m in the consistency equation by Itô's Lemma. Update θ_q^n using stochastic gradient decent.

Results: Losses

Master Equation ξ_a	Master Equation ξ_h	Goods Market	q-Drift	q-Volatility
1.02×10^{-2}	2.31×10^{-3}	3.12×10^{-4}	8.10×10^{-4}	5.57×10^{-3}

Results

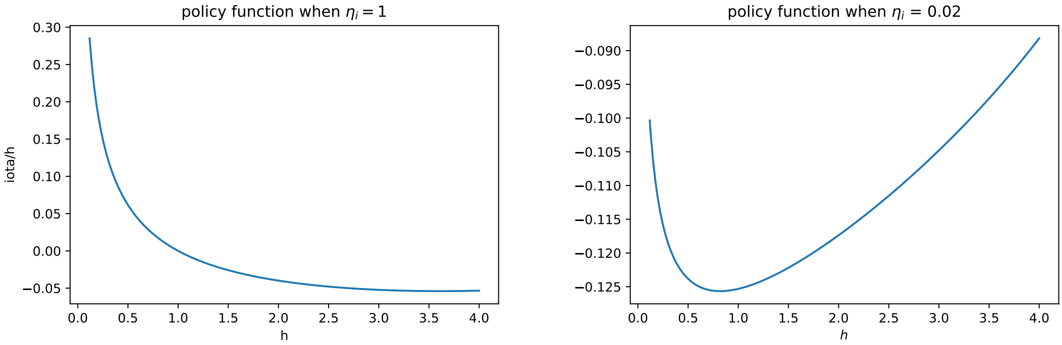


Figure: Housing Transaction Rate (when $\zeta = 1$) given liquid wealth share: 1.0(*Left*); 0.02(*Right*)

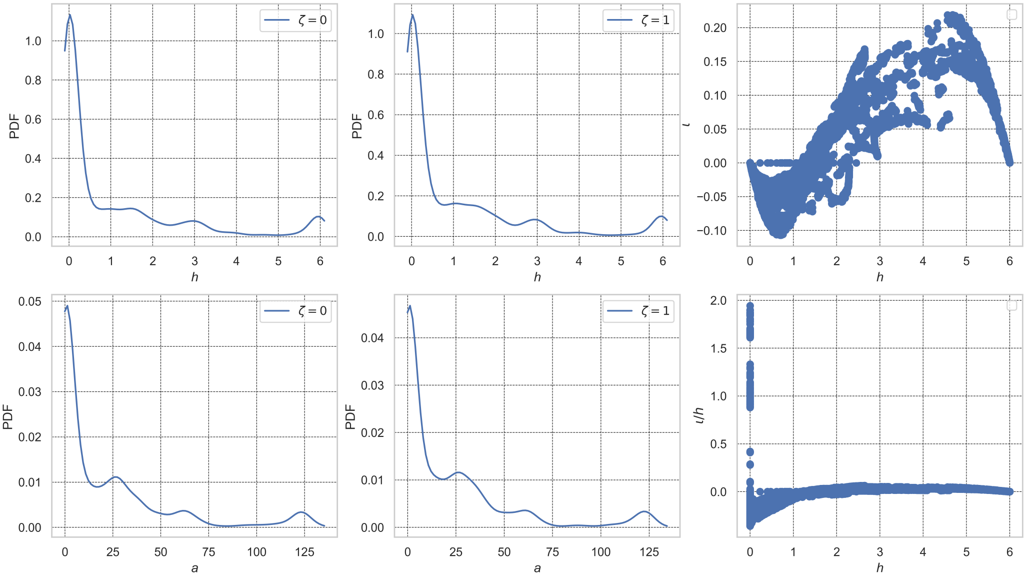
Housing and Inequality

- Revisit:

$$\iota_i = \frac{h_i}{\psi} \left(\frac{\partial V_i / \partial h}{\partial V_i / \partial a} - p \right)$$

- Decreasing utility gain from housing: negative ι for rich households
- Binded financially constraint: negative ι for poor households
- Unconstrained but lacking houses: positive ι for “mid-classes”

Results: Ergodic Distribution



Household's HJB Equation in Housing Model

$$
\begin{aligned}
\rho V_i(x_i) = \max_{b_i, c_i, \iota_i,} \; & u(c_i) \\
& + \zeta_{i,t}\nu(h_{i,t}, a_{i,t}) \frac{\partial V_i}{\partial a_i} \mu_{a_i}(b_i, c_i, \iota_i, \cdot) + \frac{\partial V_i}{\partial y} \mu^y + \lambda(\zeta_i)(V_i(a_i, h_i, \tilde{\zeta}_i, \cdot) - V_i(a_i, h_i, \zeta_i, \cdot)) \\
& + \frac{1}{2} \frac{\partial^2 V_i}{\partial a_i^2} \sigma_{a_i}^2 (b_i, \cdot) + \frac{1}{2} \frac{\partial^2 V_i}{\partial y^2} \sigma_y^2 + \frac{\partial^2 V_i}{\partial a_i \partial y} \sigma_{a_i}(b_i, \cdot) \sigma_y \\
& + \sum_{j \neq i} \frac{\partial^2 V_i}{\partial a_i \partial a_j} \sigma_{a_i}(b_i, \cdot) \hat{\sigma}_{a_j}(\cdot) + \sum_{j \neq i} \frac{\partial V_i}{\partial a_j} \hat{\mu}_{a_j}(\cdot) + \sum_{j \neq i} \frac{\partial^2 V_i}{\partial a_j \partial y} \hat{\sigma}_{a_j}(\cdot) \sigma_y \\
& + \sum_{j \neq i} \lambda(\zeta_j)(V_i(a_i, h_i, \zeta_i, \tilde{\zeta}_j \cdot) - V_i(a_i, h_i, \zeta_i, \zeta_j, \cdot)) \\
& + \frac{1}{2} \sum_{j \neq i, j' \neq i} \frac{\partial^2 V_i}{\partial a_j \partial a_{j'}} \hat{\sigma}_{a_j}(\cdot) \hat{\sigma}_{a'_j}(\cdot) + \phi(a_i, h_i, \kappa_i)
\end{aligned}
\tag{12}
$$

Back

Derivations

The first order condition of optimal portfolio choice condition in (6) can be further written into a matrix form:

$$\mathbf{M}(\boldsymbol{a}-\boldsymbol{b})=\boldsymbol{n} \tag{13}$$

By multiplying both sides with $\mathbf{M}^{-1}$, the risky asset holding can be written as:

$$\boldsymbol{a}-\boldsymbol{b}=\mathbf{M}^{-1}\boldsymbol{n} \tag{14}$$

Further, the bond market clearing condition can be essentially written as: $\boldsymbol{\iota}\cdot\boldsymbol{b}=0$, we have:

$$\boldsymbol{\iota}\cdot(\boldsymbol{a}-\boldsymbol{b})=\boldsymbol{\iota}\cdot(\mathbf{M}^{-1}\boldsymbol{n})=q. \tag{15}$$

Plug in the expression for $\boldsymbol{n}$, then we can get the expression for risk-premium.

Closed form solution for housing price p_t with quadratic transaction cost: $\Psi(h_{i,t},\iota_{i,t})=\frac{1}{2}\kappa\frac{\iota_{i,t}^2}{h_{i,t}}$

$$p+\kappa\frac{\iota_i}{h_i}=\frac{\partial V_i/\partial h_i}{\partial V_i/\partial a_i}\rightarrow p=\frac{1}{H}\left(\int_i\frac{\xi_{h,i}}{\xi_{a,i}}h_i di\right) \tag{16}$$

back