Structural Reinforcement Learning for Heterogeneous Agent Macroeconomics

Yucheng Yang*
Zurich

Chiyuan Wang*
Peking University

Andreas Schaab
Berkeley

Benjamin Moll†
LSE

*equal contribution †corresponding author

Universität Tübingen

Heterogeneous agent models with aggregate risk

- Huge literature since Krusell-Smith and Den Haan from late 90s
- Key challenge: rational expectations + general equilibrium
  $\Rightarrow$ distribution = state variable in Bellman equation ("Master equation")
  - true even though households/firms only care about prices
  - intuition: equilibrium prices are not Markov, only the distribution is
    $\Rightarrow$ forecast distributions to forecast prices
- Despite recent impressive advances to solve it directly, still lack of efficient global solution methods for advanced HA models with aggregate risk
- This paper: sidestep master eqn with structural reinforcement learning

Sidestep Master eqn using structural reinforcement learning

RL = learning value & policy functions in Markov decision processes from Monte Carlo simulation

Here: RL about equilibrium prices but not individual states $\Rightarrow$ "Structural RL"

Outcome: efficient & flexible global solution method for HA models w agg risk

- solves problems traditional methods struggle with:

1. non-trivial market clearing (Huggett w agg. risk) $\approx$ 1 min on Google Colab
2. HANK with forward-looking Phillips curve $\approx$ 3 min

How does it work?

- in contrast to dynamic programming, RL can handle non-Markov states
- replace dist'n with low-dim. prices in state space, grid-based not DNNs
- efficient market clearing using policy functions (= demand curves)

Our structural RL approach in a nutshell

- Step 1: parameterize policy function in low-dim state (s, z, p):

$$\pi_\theta(s, z, p) = \{c_\theta(s, z, p), b'_\theta(s, z, p)\},$$

  with grid values as unknown parameters θ. Do NOT parameterize price functions.

- Step 2: GE simulation given π_θ. For any θ, compute average lifetime utility

$$v(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \beta^t u(c_\theta(s_t^n, z_t^n, p_t^n))$$

  along N simulated equilibrium paths with very large T.

- Step 3: structural policy gradient. Update θ by stochastic gradient ascent on $v(\theta)$, and repeat Step 2.

Restricted perceptions equilibrium. Results so far very close to RE.

Literature and contribution

Global solution methods for HA models with aggregate risk

Fernández-Villaverde et al, Han-Yang-E, Maliar-Maliar-Winant, Azinovic-Gaegauf-Scheidegger, Schaab, Gu et al...

- sidestep Master equation rather than "taming curse of dimensionality"
- Han-Yang-E DeepHAM = also RL-inspired

Global solution with bounded rationality Krusell-Smith, Den Haan, ...

- similarity: low-dim. state space; difference: no perceived law of motion

Adaptive (least squares) learning Bray, Marcet-Sargent, Evans-Honkapohja, Jacobson, Giusto, ...

- like RL = stochastic approximation method

Self-confirming equilibrium

- agents form price exp. from data generated by economy where they live
- but expectations incorrect for off-equilibrium (or rare) price realizations
- restricted perceptions equilibrium

"Sequence space" Auclert- Bardóczy-Rognlie-Straub

- global solution in sequence space (low-dim. prices) via Monte Carlo

Plan

1. HA Models: Setup
2. The RL Approach to HA macro without the Master equation
3. Computational experiments: Krusell-Smith and Huggett with agg risk
4. Forward looking Phillips curve: HANK with aggregate shocks.

HA Models: Setup

Textbook HA model – Huggett (1993) with agg. risk

- Continuum of agents i, heterog. in $(b_{i,t}, y_{i,t})$, $y_{i,t}$ = id. risk, agg. shock z_t
- Households choose consumption $c_{i,t}$ to maximize

$$v_{i,0} = \max_{\{c_{i,t}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_{i,t}) \quad \text{subject to}$$

$$c_{i,t} + b_{i,t+1} = R_t b_{i,t} + y_{i,t} z_t, \quad y_{i,t+1} \sim \mathcal{T}_y(\cdot|y_{i,t}), \quad b_{i,t+1} \geq \underline{b}$$

- State of the economy: distribution $G_t(b, y)$ and z_t. Prices: R_t.
- Market clearing: interest rates R_t such that

$$\int b_t'(b, y)\, dG_t(b, y) = \bar{B}, \quad \text{all } t$$

Note: agent problem depends on G_t only through low-dim. prices (R_t)

General setup of HA models

- Continuum of agents i, heterog. in $s \in \mathbb{R}^n$, e.g. wealth, labor prod'ty
- State of the economy: distribution $G_t(s)$ and $z_t \in \mathbb{R}^k$. Prices $p_t \in \mathbb{R}^\ell$.
- Agents choose consumption $c_{i,t}$ to maximize

$$v_{i,0} = \max_{\{c_{i,t}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_{i,t}) \quad \text{subject to}$$

$$s_{i,t+1} \sim \mathcal{T}_s(\cdot | s_{i,t}, c_{i,t}, p_t, z_t) = \text{budget constraint + income process}$$

- Low-dimensional equilibrium price functionals

$$p_t = P^*(G_t, z_t), \qquad z_{t+1} \sim \mathcal{T}_z(\cdot | z_t)$$

P^* can be implicit e.g. Huggett, or analytical, e.g. Krusell-Smith 98.

Note: agent problem depends on G_t only through low-dim. price functionals

Discretized representation

- Discretize individual state $s \in \{s_1, ..., s_J\}$ with $J = J_1 \times ... \times J_n$
- Value function, distribution, etc are J-dimensional vectors

$$\mathbf{v}_t = \begin{bmatrix} v_t(s_1) \\ \vdots \\ v_t(s_J) \end{bmatrix}, \qquad \mathbf{g}_t = \begin{bmatrix} g_t(s_1) \\ \vdots \\ g_t(s_J) \end{bmatrix}$$

- Consumption policy $c = \pi_t(s, z) \Rightarrow J \times J$ transition matrix for s

$$\mathbf{A}_{\pi_t(z_t)} \quad \text{with entries} \quad \Pr(s_{j'}|s_j) = \mathcal{T}_s(s_{j'}|s_j, \pi_t(s_j, z_t), p_t, z_t)$$

- High-dimensional state $(\mathbf{g}_t, z_t)$ is Markov:

$$\mathbf{g}_{t+1} = \mathbf{A}_{\pi_t(z_t)}^\top \mathbf{g}_t, \quad z_{t+1} \sim \mathcal{T}_z(\cdot|z_t)$$

- Low-dim equilibrium prices $p_t = P^*(\mathbf{g}_t, z_t)$: not Markov

Key difficulty: equilibrium prices are not Markov

- Low-dim equilibrium prices $p_t = P^*(\mathbf{g}_t, z_t)$: not Markov
- ... only extremely high-dimensional $(\mathbf{g}_t, z_t)$ is
- Dynamic programming can only handle Markov states $\Rightarrow$ Master equation

$$V(s, \mathbf{g}, z) = \max_c u(c) + \beta \mathbb{E}\left[V(s', \mathbf{g}', z') | s, \mathbf{g}, z\right] \text{ s.t. } s' \sim \mathcal{T}_s(\cdot | s, c, P^*(\mathbf{g}, z))$$

- Without Markov transition prob's: cannot even write Bellman equation!
- Our solution: use RL to approximate value/policy functions with low-dim non-Markov state variables

Sidestepping the Master Equation via Structural RL

What is reinforcement learning? A simple example

RL = learning value & policy functions from Monte Carlo ‣ RL Primer

Example: compute value function (eliminate actions & individual states for now)

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \beta \int v(p')f(p'|p)dp'$$

What is reinforcement learning? A simple example

RL = learning value & policy functions from Monte Carlo RL Primer

Example: compute value function (eliminate actions & individual states for now)

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \beta \int v(p') f(p'|p) dp'$$

2. RL/Monte Carlo: don't know f but sample N trajectories $\{p_t^i\}_{t=0}^{T}$

$$v_0 \approx \widehat{v}_0 = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \beta^t u(p_t^i) \quad \text{or} \quad \widehat{v}_0^k = \widehat{v}_0^{k-1} + \frac{1}{k}\left(\sum_{t=0}^{T} \beta^t u(p_t^k) - \widehat{v}_0^{k-1}\right)$$

What is reinforcement learning? A simple example

RL = learning value & policy functions from Monte Carlo ▸ RL Primer

Example: compute value function (eliminate actions & individual states for now)

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \beta \int v(p') f(p'|p) dp'$$

2. RL/Monte Carlo: don't know f but sample N trajectories $\{p_t^i\}_{t=0}^{T}$

$$v_0 \approx \widehat{v}_0 = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \beta^t u(p_t^i) \quad \text{or} \quad \widehat{v}_0^k = \widehat{v}_0^{k-1} + \frac{1}{k}\left(\sum_{t=0}^{T} \beta^t u(p_t^k) - \widehat{v}_0^{k-1}\right)$$

Can also be extended to compute optimal policy: policy gradient method

Sidestepping the Master Equation via Structural RL

Recall: states $s = (b, y)$, prices $p = (R)$, agents choose $c_{i,t}$ to maximize

$$v_{i,0} = \max_{\{c_{i,t}\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(c_{i,t})\right] \text{ s.t. } s_{i,t+1} \sim \mathcal{T}_s(\cdot|s_{i,t}, c_{i,t}, p_t, z_t), \quad p_t = P^*(G_t, z_t) \qquad (1)$$

Assumption 1: agents observe prices p_t but not distribution $G_t(s)$ ▸ Wold repr.

- p_t can include moments of G_t, say GDP, as long as low-dimensional

Similarity to standard RL: don't know transition prob. of prices p_t

Difference to standard RL: know "local environment" $\mathcal{T}_s$ and u (Han-Yang-E)

Assumption 2: consumption policy π does not condition on price histories

$$c_{i,t} = \pi(s_{i,t}, p_t, z_t)$$

- To do: keep track of price histories, e.g. h lags or RNN

RL: optimize $\pi(s_{i,t}, p_t, z_t)$ to solve (1) on the simulated paths. ▸ RL Primer

Restricted perceptions equilibrium

A pair of mappings (π^*, P^*) constitutes a restricted perceptions equilibrium if:

1. Optimality. For any price sequence $\{p_t\}$ generated by $p_t = P^*(\mathbf{g}_t, z_t)$ and exogenous sequence $\{z_t\}$, agents choose $\pi^*(s, p, z)$ to solve:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(\pi(s_t, p_t, z_t))\right],$$

   subject to the individual budget constraint and state transition equations.

2. Market clearing. For every period t, all markets clear.

3. Consistency. The prices that agents use to form expectations coincide with the prices in the simulated economy when all agents follow π^*:

$$p_t = P^*(\mathbf{g}_t, z_t), \qquad \mathbf{g}_{t+1} = \mathbf{A}^{\top}_{\pi^*(p_t, z_t)} \mathbf{g}_t$$

Simulating the economy given policy $c = \pi(s, p, z)$

For given (suboptimal) policy $\pi(s, p, z)$, can simulate economy forward in time

- important: very cheap computationally with JAX on GPUs

Recall: discrete $s \Rightarrow$ vectors $\boldsymbol{\pi}(p, z)$, $\mathbf{g}_t$, sparse transition matrix $\mathbf{A}_{\pi(p,z)}$

For given policy $\boldsymbol{\pi}(p, z)$ and $(\mathbf{g}_0, z_0)$, economy evolves as:

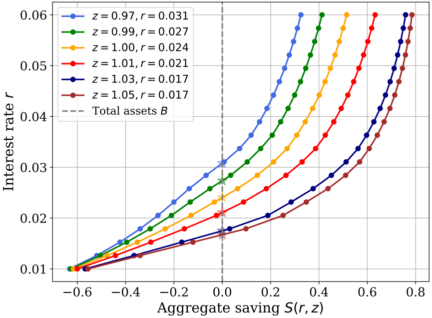
$$
\begin{aligned}
p_t &= P^*(\mathbf{g}_t, z_t) \\
\mathbf{g}_{t+1} &= \mathbf{A}^\top_{\pi(p_t, z_t)} \mathbf{g}_t \\
z_{t+1} &\sim \mathcal{T}_z(\cdot | z_t)
\end{aligned}
$$

Efficient handling of non-trivial market clearing

$$S_t(p, z) = \bar{B}, \qquad S_t(p, z) := \int b'(s, p, z)\, dG_t(s) = \text{agg. saving supply}$$

Key: integrate policies $b'(s, p, z) \Rightarrow$ aggregate saving on the price grid $S(p, z_t)$

$$\Rightarrow p_t \;\; \text{solves} \;\; S_t(p_t, z_t) = \mathbf{b}'(p_t, z_t)^{\top} \mathbf{g}_t = \bar{B}$$



Market clearing is part of environment, not another loop!

Using knowledge of local environment

Value function for given policy $\pi(s, p, z)$

$$v_\pi(s, p, z) = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(\pi(s_{i,t}, p_t, z_t)) \middle| s_{i,0} = s, p_0 = p, z_0 = z\right] \qquad (*)$$

Partition state space (s, p, z) into known dynamics and unknown dynamics

- use transition matrix $\mathbf{A}$ to keep track of all s-transitions
- expectation $\mathbb{E}$ only over price trajectories $\{p_t\}_{t=0}^{\infty}$ and $\{z_t\}_{t=0}^{\infty}$

Write $v_\pi(s, p, z)$ in $(*)$ as vector $\mathbf{v}_\pi(p, z)$:

$$\mathbf{v}_\pi(p, z) = \mathbb{E}\left[\mathbf{u}_0 + \beta \mathbf{A}_0 \mathbf{u}_1 + \beta^2 \mathbf{A}_0 \mathbf{A}_1 \mathbf{u}_2 + \ldots | p, z\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \mathbf{A}_{0\to t} \mathbf{u}_t \middle| p, z\right]$$

where $\mathbf{u}_t = u(\boldsymbol{\pi}(p_t, z_t))$ and $\mathbf{A}_t = \mathbf{A}_{\pi(p_t, z_t)}$ and $\mathbf{A}_{0\to t} = \mathbf{A}_0 \cdots \mathbf{A}_{t-1}$

Summary: problem to be solved

Find optimal policy $\pi(s, p, z)$ or $\boldsymbol{\pi}(p, z)$ that maximizes

$$\mathbf{v}_\pi(p, z) = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \mathbf{A}_{\pi,0\to t} u(\boldsymbol{\pi}(p_t, z_t)) \middle| p_0, z_0 = p, z\right]$$

taking as given evolution of equilibrium prices p_t (agent does not use P^*)

$$p_t = P^*(\mathbf{g}_t, z_t), \quad \mathbf{g}_{t+1} = \mathbf{A}^\top_{\tilde{\pi}(p_t, z_t)} \mathbf{g}_t, \quad z_{t+1} \sim \mathcal{T}_z(\cdot | z_t), \quad t = 0, 1, \ldots$$

with $(\mathbf{g}_0, z_0)$ given, $\tilde{\boldsymbol{\pi}} = \boldsymbol{\pi}$ in eqm, and $\mathbf{A}_{\pi,0\to t} = \mathbf{A}_{\pi(p_0,z_0)} \cdots \mathbf{A}_{\pi(p_{t-1},z_{t-1})}$

Key observation:

- State of economy = $(\mathbf{g}, z)$ = very high-dimensional
- But state in value/policy functions = (s, p, z) = very low-dimensional!
- No perceived law of motion, inner loop / outer loop (like in Krusell-Smith)
- GE problem only mildly more difficult than PE

RL policy gradient method for maximizing $\mathbf{v}_\pi$

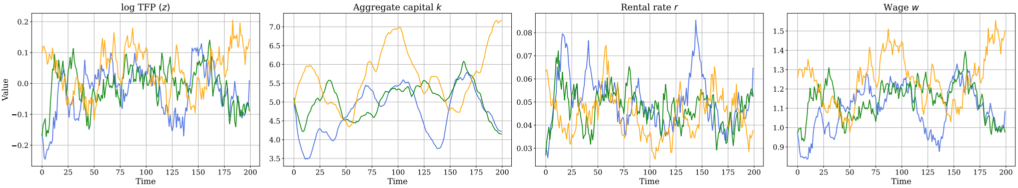
Find optimal policy $\boldsymbol{\pi}(p, z)$ that maximizes estimate of $\mathbb{E}_{p_0 \sim \psi_p, z_0 \sim \psi_z}[\mathbf{v}_\pi(p_0, z_0)]$:

$$\widehat{\mathbf{v}}_\pi = \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \beta^t \mathbf{A}_{\pi, 0 \to t} u(\boldsymbol{\pi}(p_t^i, z_t^i)) \right]$$

with N price trajectories p_t^i sampled from interacting with environment (rollouts):

$$p_t = P^*(\mathbf{g}_t, z_t), \quad \mathbf{g}_{t+1} = \mathbf{A}^{\top}_{\tilde{\pi}(p_t, z_t)} \mathbf{g}_t, \quad z_{t+1} \sim \mathcal{T}_z(\cdot | z_t), \quad t = 0, 1, \ldots$$

with $\mathbf{g}_0 \sim \psi_g(\cdot)$, $z_0 \sim \psi_z(\cdot)$ and with $\tilde{\boldsymbol{\pi}} = \boldsymbol{\pi}$ in equilibrium



RL policy gradient method for maximizing $\mathbf{v}_\pi$

Find optimal policy $\boldsymbol{\pi}(p, z)$ that maximizes estimate of $\mathbb{E}_{p_0 \sim \psi_p, z_0 \sim \psi_z}[\mathbf{v}_\pi(p_0, z_0)]$:

$$\widehat{\mathbf{v}}_\pi = \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \beta^t \mathbf{A}_{\pi, 0 \to t} u(\boldsymbol{\pi}(p_t^i, z_t^i)) \right]$$

with N price trajectories p_t^i sampled from interacting with environment (rollouts):

$$p_t = P^*(\mathbf{g}_t, z_t), \quad \mathbf{g}_{t+1} = \mathbf{A}^\top_{\tilde{\pi}(p_t, z_t)} \mathbf{g}_t, \quad z_{t+1} \sim \mathcal{T}_z(\cdot | z_t), \quad t = 0, 1, \ldots$$

with $\mathbf{g}_0 \sim \psi_g(\cdot)$, $z_0 \sim \psi_z(\cdot)$ and with $\tilde{\boldsymbol{\pi}} = \boldsymbol{\pi}$ in equilibrium

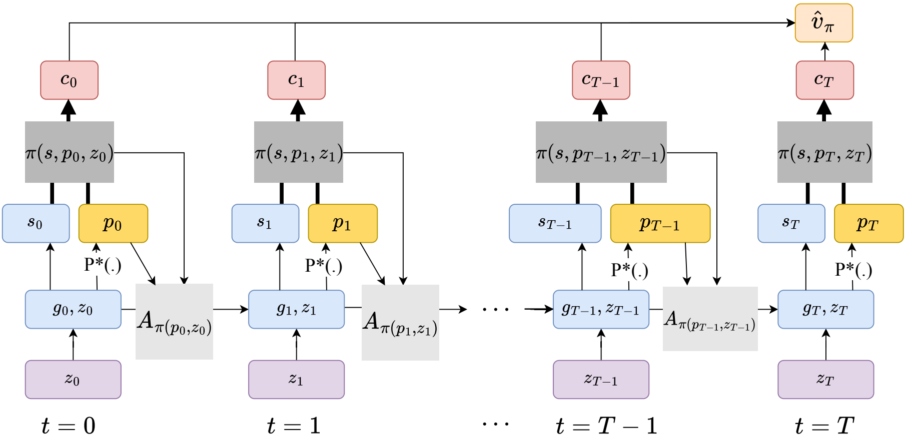
In practice, maximize scalar objective using gradient ascent:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbf{d}_0^\top \widehat{\mathbf{v}}_\pi = \sum_{j=1}^{J} d_0(s_j) \widehat{v}_\pi(s_j), \quad d_0(s) = \text{uniform dist. over } s$$

policy either grid based $\boldsymbol{\theta} = [\boldsymbol{\pi}(p_1, z_1), \ldots, \boldsymbol{\pi}(p_J, z_K)]$ or neural net $\pi(s, p, z; \boldsymbol{\theta})$

- low dim. $\Rightarrow$ grid-based method works well so far, no need for neural nets!

Computational graph for construction of $\widehat{\mathbf{v}}_\pi$



Computational experiments

Runtimes

- Efficient implementation in JAX for GPUs, run on Google Colab
- Stochastic algorithm: present averages over multiple runs

Model	Average converge epoch	# Runs	Average Runtime (sec)
Krusell-Smith	438.4	10	56.55
Huggett with agg. shocks	480.6	10	75.29
HANK with agg. shocks	496.5	10	199.53
Partial equilibrium (Huggett)	289.3	10	39.49

Note: all experiments were implemented on the A100 GPU on Google Colab

Krusell-Smith model

Computational experiments: Krusell-Smith model

Parameter	Description	Value
α	Capital share	0.36
δ	Capital depreciation rate	0.08
γ	Discount factor	0.95
σ	Coefficient of relative risk aversion	3
ρ_z	Persistence of AR(1) for z_t (log TFP)	0.9
ν_z	Volatility of AR(1) for z_t (log TFP)	0.03

Hyperparameter	Description	Value
J_{s_1}	Number of s_1 (wealth) grid points	200
J_{s_2}	Number of s_2 (income) states	3
J_{p_1}	Number of p_1 (rental rate) grid points	50
J_{p_2}	Number of p_2 (wage) grid points	70
N	Sample size = number of p trajectories	256,512,1024,...
T	Time truncation s.t. $\beta^T < 0.01$	90
ϵ	Convergence criterion on $\widehat{\mathbf{v}}_\pi$	0.001
$\eta_{\mathtt{ini}}$	Initial learning rate	0.01
$\eta_{\mathtt{decay}}$	Learning rate decay (exponential)	0.5

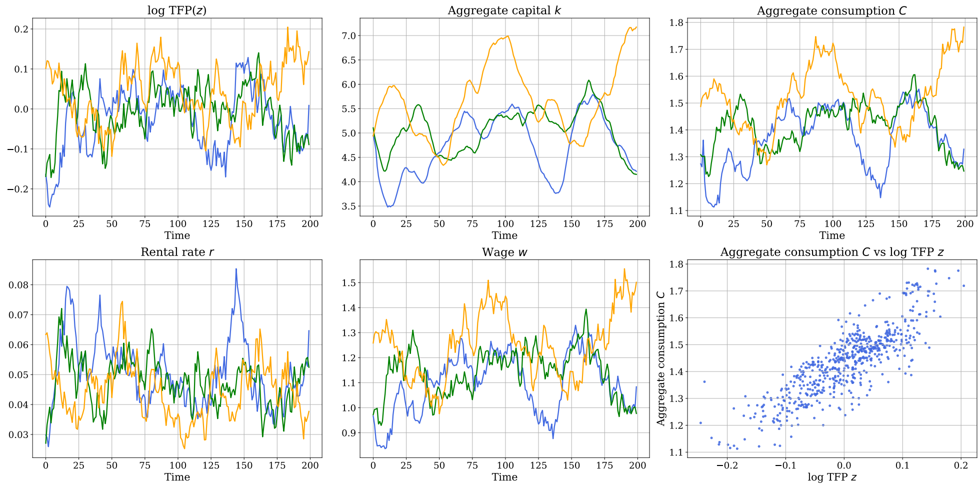
Runtimes

- Runtime increases with sample size N
- GE problem only mildly more difficult than PE

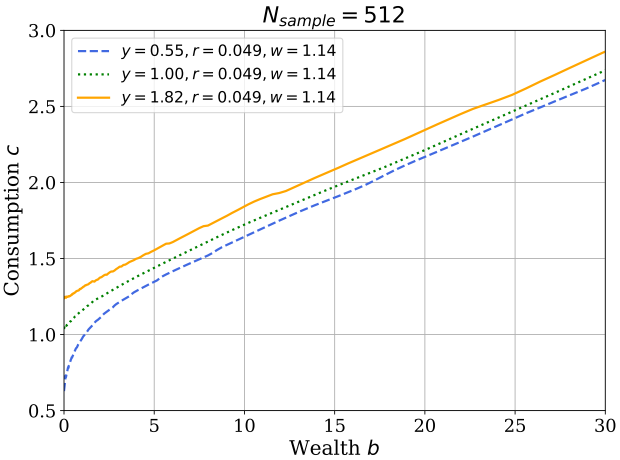
Sample size N	PE/GE	Average converge epoch	# Runs	Average Runtime (sec)
256	GE	319.3	10	27.92
512	GE	304.3	10	40.39
1024	GE	357.6	10	81.02
2048	GE	384.8	10	160.08
512	PE	586.0	10	41.44

Note: all experiments were implemented on the A100 GPU on Google Colab

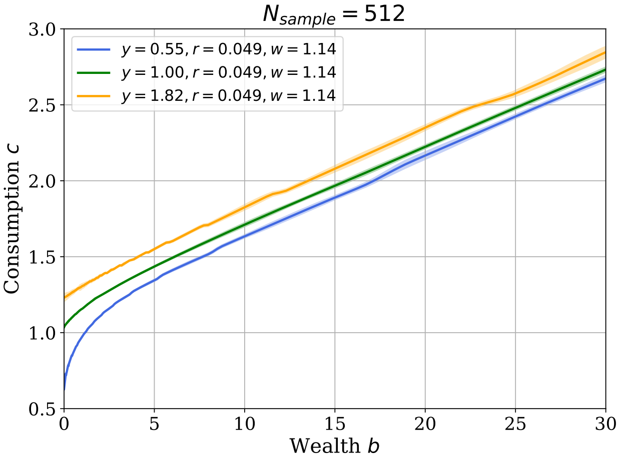
Some simulated trajectories under the optimal policy



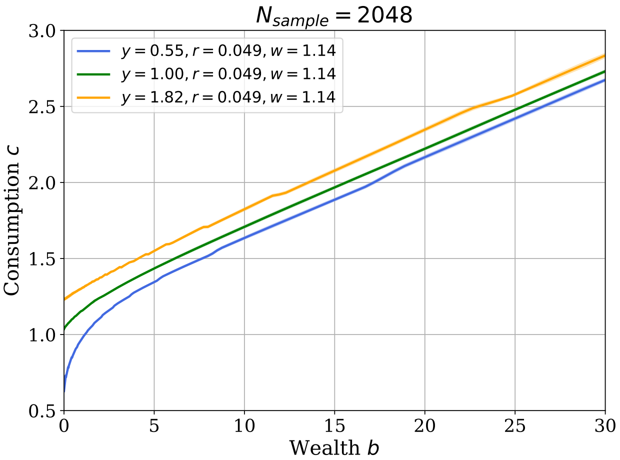
Consumption policy function: single run



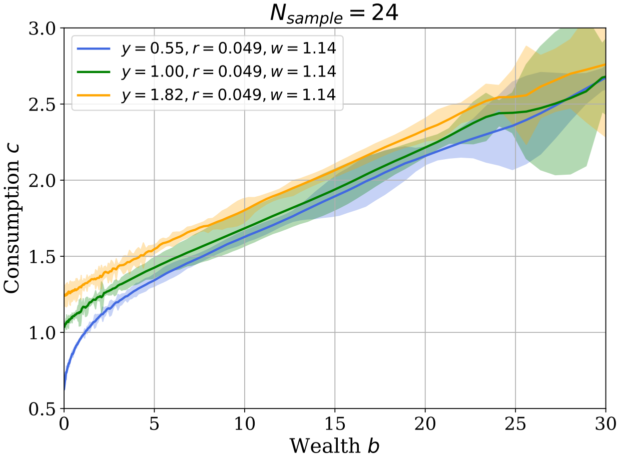
Consumption policy function: multiple runs



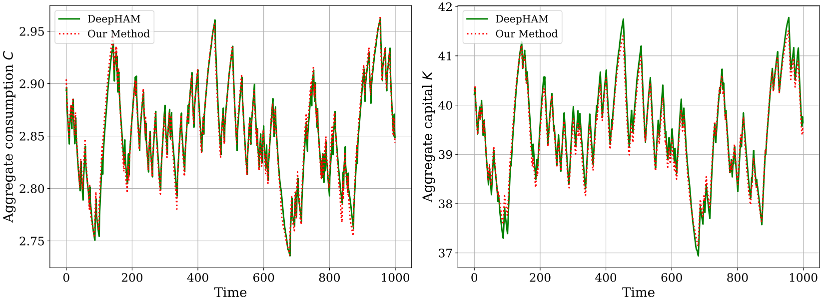
Larger sample size $N \Rightarrow$ more precise estimate



Smaller sample size $N \Rightarrow$ noisier estimate



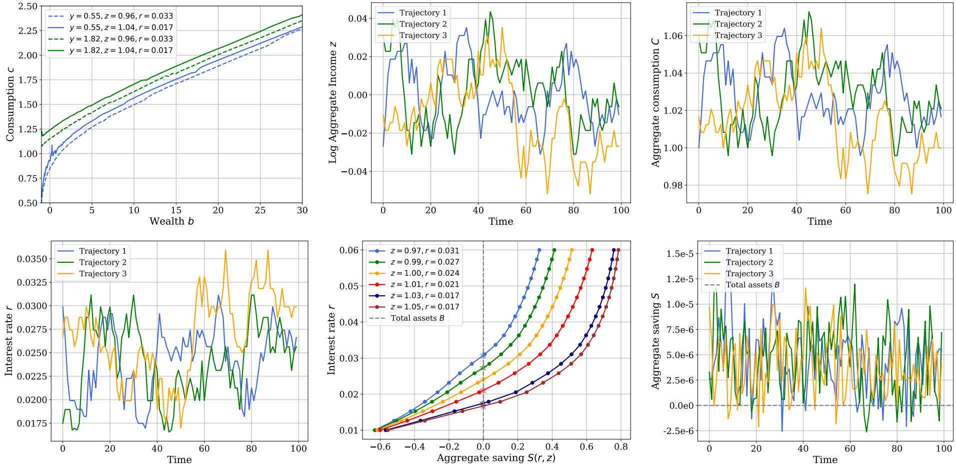
Structural RL method recovers RE solutions



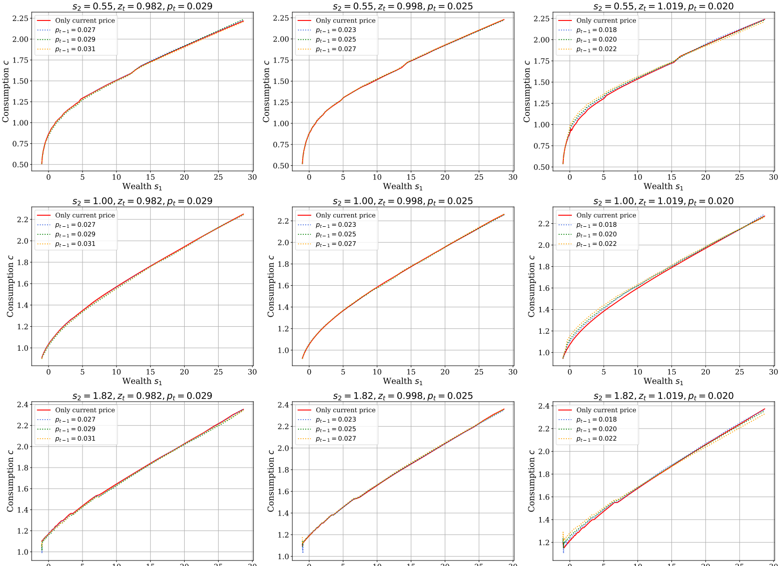
RE solutions are obtained with a deep learning based method (DeepHAM).

Non-trivial market clearing (Huggett)

Some simulated trajectories under the optimal policy



Adding one lagged price p_{t-1} into state space



HANK with forward-looking Phillips curve

HANK with forward-looking Phillips curve

Household block (similar to before): states $s = (b, y)$ and prices $p = (R, w)$

$$\text{policies} = \{c(s,p), n(s,p)\} \quad \text{that maximize PDV of utility}$$

Firm block: price setting $\Rightarrow$ forward-looking Phillips curve = added difficulty

$$\Pi_t = \frac{\varepsilon}{\theta}\left(\frac{w_t}{z_t} - m^*\right) + \mathbb{E}\left[R_{t+1}^{-1}\frac{Y_{t+1}}{Y_t}\Pi_{t+1}\middle|\mathcal{I}_t\right], \qquad m^* = \frac{\varepsilon - 1}{\varepsilon}$$

Conventional approach: parameterize $\mathbb{E}[\Pi_{t+1}|\mathcal{I}_t] \Rightarrow$ complicated fixed-point
(e.g. Kase-Melosi-Rottner, Fernández-Villaverde et al)

HANK with forward-looking Phillips curve

Household block (similar to before): states $s = (b, y)$ and prices $p = (R, w)$

$$\text{policies} = \{c(s,p), n(s,p)\} \quad \text{that maximize PDV of utility}$$

Firm block: price setting $\Rightarrow$ forward-looking Phillips curve = added difficulty

Our solution: solve firm price-setting problem using policy gradient method

$$J_0 = \max_{\{P_{j,t}\}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} R_{0\to t}^{-1} \left\{ \text{Profits}\left(\frac{P_{j,t}}{P_t}, \frac{w_t}{z_t}, Y_t \right) - \frac{\theta}{2} \left(\frac{P_{j,t} - P_{j,t-1}}{P_{j,t-1}} \right)^2 \right\} \right]$$

HANK with forward-looking Phillips curve

Household block (similar to before): states $s = (b, y)$ and prices $p = (R, w)$

$$\text{policies} = \{c(s,p), n(s,p)\} \quad \text{that maximize PDV of utility}$$

Firm block: price setting $\Rightarrow$ forward-looking Phillips curve = added difficulty

Or in terms of inflation $\Pi_t = (P_t - P_{t-1})/P_t$

$$J_0 = \max_{\{\Pi_{j,t}\}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} R_{0\to t}^{-1} \left\{ \text{Profits}\left(\frac{1+\Pi_{j,t}}{1+\Pi_t}, \frac{w_t}{z_t}, Y_t \right) - \frac{\theta}{2} \left(\Pi_{j,t}\right)^2 \right\} \right]$$

HANK: policy gradient method for both households & firms

Household block (similar to before): states $s = (b, y)$ and prices $p = (R, w)$

$$\text{policies} = \{c(s,p), n(s,p)\} \quad \text{that maximize PDV of utility}$$

Firm block: states z and prices $p = (R, w)$

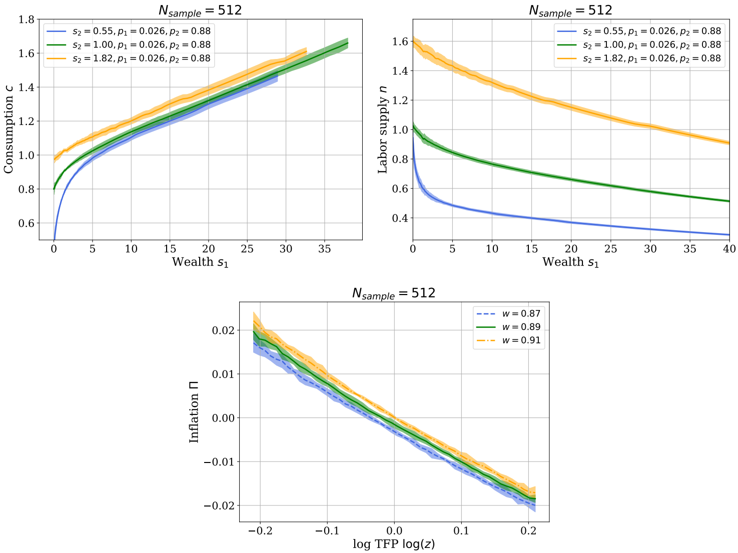
$$\text{policy} = \Pi(z,p) \quad \text{that maximizes}$$

$$J_\Pi = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} R_{0\to t}^{-1} \left\{ \text{Profits}\left(\frac{1+\Pi(z_t, R_t, w_t)}{1+\Pi_t}, \frac{w_t}{z_t}, Y_t \right) - \frac{\theta}{2} \left(\Pi(z_t, R_t, w_t)\right)^2 \right\} \right]$$

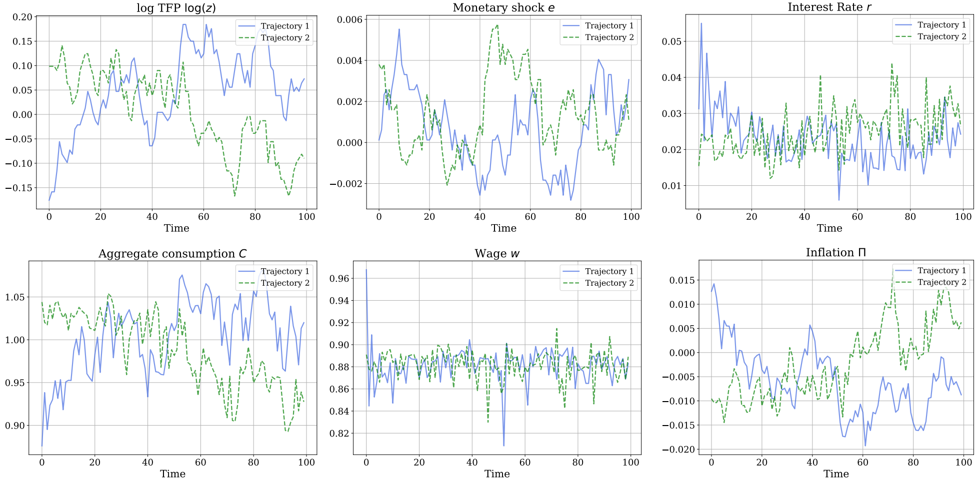
Symmetric treatment of firms and households, update policies simultaneously

In practice: good convergence properties

HANK: Household and firm policy functions



HANK simulations



Summary

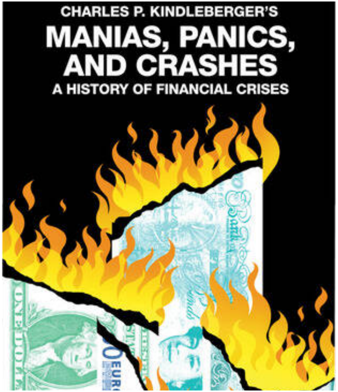
Efficient and flexible global solution method for non-stationary HA models

Reinforcement learning about equilibrium prices (but not individual states)

- sidestep infinite-dimensional Master equation
- solve much lower-dimensional problem

Solves problems traditional methods struggle with

- non-trivial market clearing conditions
- HANK with forward-looking Phillips curve
- next: models of large crises, booms/busts



Thanks!

In stationary world, lagged prices are enough for RE ▸ back

Recall Assumption 1: agents observe prices p_t but not distribution $G_t(s)$

Important: Assumption 1 still consistent with rational expectations

Why? Wold representation theorem!

Step 1 (Wold): if p_t-process is stationary, it has Wold representation = VMA(∞)

$$p_t = \sum_{j=0}^{\infty} c_j \varepsilon_{t-j}, \qquad c_j = \text{some unknown coefficients}$$

Step 2: if VMA(∞) is invertible, it can be expressed as a VAR(∞) and hence

$$p_{t+1} \sim \mathcal{T}_p(\cdot | p_t, p_{t-1}, \ldots)$$

In practice, include finitely many lags

Assumption 2: extreme case with zero lags $p_{t+1} \sim \mathcal{T}_p(\cdot | p_t)$

Key difficulty: equilibrium prices are not Markov

- Equilibrium prices satisfy back

$$
\begin{aligned}
p_t &= P^*(\mathbf{g}_t, z_t) \\
\mathbf{g}_{t+1} &= \mathbf{A}^\top_{\pi_t(z_t)} \mathbf{g}_t \\
z_{t+1} &\sim \mathcal{T}_z(\cdot | z_t)
\end{aligned}
$$

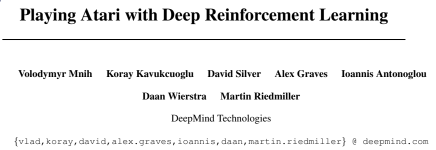
- Difficulty: low-dimensional p_t does not have Markov property...
- ... only extremely high-dimensional $(\mathbf{g}_t, z_t)$ does
- Dynamic programming can only handle Markov states $\Rightarrow$ Master equation

$$V(s, \mathbf{g}, z) = \max_c u(c) + \beta \mathbb{E}\left[V(s', \mathbf{g}', z') | s, \mathbf{g}, z\right] \text{ s.t. } s' \sim \mathcal{T}_s(\cdot | s, c, P^*(\mathbf{g}, z))$$

- Without Markov transition prob's: cannot even write Bellman equation!
- But what if there was a way to approximate value and policy functions with p_t process for which there are no Markov transition probabilities?

Brief primer on reinforcement learning

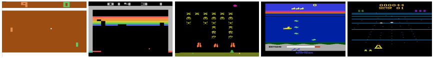
RL: learning value & policy functions in incompletely-known Markov decision processes from experience (Monte Carlo sampling) a.k.a. “approximate DP”

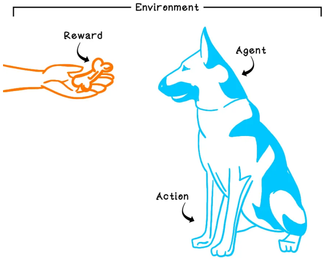

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com





Computing an expected value

Random variable x

How compute expected value $\mathbb{E}[x]$? Two approaches:

1. Exact: know probability distribution $f(x) \Rightarrow$ calculate

$$\mathbb{E}[x] = \int x f(x) dx$$

2. Monte Carlo: don't know f but can sample $\{x_1, x_2, \ldots, x_N\}$

$$\mathbb{E}[x] \approx \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Or update incrementally (stochastic approximation method):

$$\bar{x}_k = \frac{1}{k} \sum_{i=1}^{k} x_i \quad \text{satisfies} \quad \bar{x}_k = \bar{x}_{k-1} + \frac{1}{k}(x_k - \bar{x}_{k-1}), \quad \frac{1}{k} = \text{“learning rate”}$$

Computing a value function

For now: eliminate actions and individual states

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \int v(p')f(p'|p)dp'$$

Computing a value function

For now: eliminate actions and individual states

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \int v(p')f(p'|p)dp'$$

2. Monte Carlo: don't know f but sample N trajectories $\{p_t^i\}_{t=0}^T$

$$v_0 \approx \widehat{v}_0 = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\beta^t u(p_t^i) \quad \text{or} \quad \widehat{v}_0^k = \widehat{v}_0^{k-1} + \frac{1}{k}\left(\sum_{t=0}^{T}\beta^t u(p_t^k) - \widehat{v}_0^{k-1}\right)$$

Computing a value function

For now: eliminate actions and individual states

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \int v(p')f(p'|p)dp'$$

2. Monte Carlo: don't know f but sample N trajectories $\{p_t^i\}_{t=0}^{T}$

$$v_0 \approx \hat{v}_0 = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T} \beta^t u(p_t^i) \quad \text{or} \quad \hat{v}_0^k = \hat{v}_0^{k-1} + \frac{1}{k}\left(\sum_{t=0}^{T} \beta^t u(p_t^k) - \hat{v}_0^{k-1}\right)$$

Can also be extended to compute optimal policy: policy gradient method

Computing a value function

For now: eliminate actions and individual states

$$v_0 = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(p_t)\right], \quad p_t = \text{exogenous stochastic process}$$

Two approaches:

1. Dynamic programming: p_t Markov and know $f(p'|p)$

$$v(p) = u(p) + \int v(p') f(p'|p) dp'$$

2. Temporal difference learning: N trajectories, update v incrementally

$$\widehat{v}^k(p_t^k) = \widehat{v}^{k-1}(p_t^k) + \frac{1}{k}\left(\left[\sum_{\tau=0}^{n-1} \beta^\tau u(p_{t+\tau}^k) + \beta^n \widehat{v}^{k-1}(p_{t+n}^k)\right] - \widehat{v}^{k-1}(p_t^k)\right)$$

Can also be extended to compute optimal policy: policy gradient method

Agent vs Environment, Rollout of a Policy

RL distinguishes between agent and environment = everything outside of agent

In HA macro:

- agents = households and their policies
- environment = everything else, including market clearing etc

Related: rollout = agent interacting with environment under given policy

- take any (generally suboptimal) policy, run it forward in time
- how optimize policy is completely separate question (policy improvement)

This viewpoint will be important, in particular for non-trivial market clearing

back