Lecture 6: Endogenous Grid Method (EGM)

Yasuyuki Sawada, Yaolang Zhong

University of Tokyo

yaolang.zhong@e.u-tokyo.ac.jp

November 4, 2025

Recap: Households' Dynamic Optimization Problem

- The representative household chooses $\{c_t, k_{t+1}\}$ to maximize lifetime utility:

$$\max_{\{c_t, k_{t+1}\}_{t \geqslant 0}} \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} = R_t k_t + w_t, \quad k_{t+1} \geqslant b.$$

  where $R_t = 1 + r_t$ and $P(w_{t+1}|w_t)$ is taken as given to agent

- State k_t, w_t determine the income
- Controls c_t, k_{t+1} determine the expenditure. We pick $c_t = \pi(k_t, w_t)$ and then $k_{t+1} = R_t k_t + w_t - \pi(k_t, w_t)$
- Recursively (drop index t), we derived the Euler equation in the case of not binding borrowing constraint:

$$u_c(c) = \beta R \mathbb{E}[u_c(c')] = \beta R \int_{w'} u_c(c') P(dw' \mid w) \quad \text{if } k' \geqslant b$$

Recap: Dynamic Optimization as a KKT Problem

- $k' = Rk + w - \pi(k,w) \geqslant b$ can be rewritten as $\pi(k,w) \leqslant Rk + w - b \equiv \bar{\pi}(k,w)$.
- Replacing c with $\pi(k,w)$ and inverting u_c gives

$$\pi(k,w) = \min\left\{u_c^{-1}\Big[\beta R \int_{w'} u_c\big(\pi(k',w')\big)\, P(dw' \mid w)\Big],\ \bar{\pi}(k,w)\right\}$$
$$\text{s.t.} \quad k' = Rk + w - \pi(k,w).$$

  The policy π^* that satisfies the above is the solution.
- Time Iteration (TI): define the Coleman operator H so that, for a policy $\pi^{(i)}$,

$$\pi^{(i+1)}(k,w) = H(\pi^{(i)})(k,w) = \min\left\{u_c^{-1}\Big[\beta R \int_{w'} u_c\big(\pi^{(i)}(k',w')\big)\, P(dw' \mid w)\Big],\ \bar{\pi}(k,w)\right\}$$
$$\text{s.t.} \quad k' = Rk + w - \pi^{(i+1)}(k,w).$$

  The fixed point $\pi^* = H(\pi^*)$ is the desired policy.
- For a particular (k,w), $\pi^{(i+1)}(k,w)$ is solved implicitly because $k' = Rk + w - \pi^{(i+1)}(k,w)$ feeds back into the RHS of the Euler equation.

Time Iteration (TI): Implementation

- Represent the policy on a tensor grid: $\bar{\mathbf{k}} = \{\bar{k}_1, \ldots, \bar{k}_M\}$ and $\bar{\mathbf{w}} = \{\bar{w}_1, \ldots, \bar{w}_N\}$. Store the policy function $\pi^{(i)}$ in each iteration as a consumption array $\mathbf{c}^{(i)} \in \mathbb{R}^{M \times N}$ with $\pi^{(i)}(\bar{k}_m, \bar{w}_n) := c^{(i)}_{m,n}$.
- The saving array $\mathbf{k}'^{(i)}$ can be obtained by $k'^{(i)}_{m,n} = R\,\bar{k}_m + \bar{w}_n - c^{(i)}_{m,n}$
- View the policy as parameterized by the grids and coefficients:

$$\pi^{(i)}(k, w) \equiv \pi^{(i)}\big(k, w;\, \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)}\big).$$

- For $\bar{k}_m \leqslant k \leqslant \bar{k}_{m+1}$ at fixed $\bar{w}_n$, use linear interpolation

$$\lambda(k) := \frac{k - \bar{k}_m}{\bar{k}_{m+1} - \bar{k}_m} \in [0, 1], \qquad \pi^{(i)}(k, \bar{w}_n) = (1 - \lambda(k))\, c^{(i)}_{m,n} + \lambda(k)\, c^{(i)}_{m+1,n}.$$

  This extends to bilinear interpolation over (k, w) (e.g., midpoints in w) and to higher-order schemes for smoothness.

Time Iteration (TI): Implementation

- The Coleman operator $H : \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N}$ maps $\mathbf{c}^{(i)}$ to $\mathbf{c}^{(i+1)} = H(\mathbf{c}^{(i)})$ by solving a nonlinear $M \times N$–dimensional system of implicit scalar equations (one per $(\bar{k}_m, \bar{w}_n)$):

$$
\begin{aligned}
c_{m,n}^{(i+1)} &= \min\left\{u_c^{-1}\left(\beta R \sum_{n'=1}^{N} P_{nn'}\, u_c\Big(\pi^{(i)}\big(k_{m,n}'^{(i+1)},\ \bar{w}_{n'};\ \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)}\big)\Big)\right),\ \bar{\pi}_{m,n}\right\}, \\
k_{m,n}'^{(i+1)} &:= R\,\bar{k}_m + \bar{w}_n - c_{m,n}^{(i+1)}, \\
\bar{\pi}_{m,n} &:= R\,\bar{k}_m + \bar{w}_n - b.
\end{aligned}
$$

- Here w' lives on the same grid $\bar{\mathbf{w}}$; the sum uses the next-period index n' with transition weights $P_{nn'}$ (from $\bar{w}_n$ to $\bar{w}_{n'}$).
- This is a nonlinear implicit equation for $c_{m,n}^{(i+1)}$: the unknown appears on both sides; the update is also piecewise due to the min with $\bar{\pi}_{m,n}$.
- $\pi^{(i)}\big(k_{m,n}'^{(i+1)}, \bar{w}_{n'}\big)$ is evaluated by interpolating over the k-grid $\bar{\mathbf{k}}$ at each fixed $\bar{w}_{n'}$.

Endogenous Grid Method (EGM)

- Keep the grids $\bar{\mathbf{k}} = \{\bar{k}_1, \ldots, \bar{k}_M\}$ and $\bar{\mathbf{w}} = \{\bar{w}_1, \ldots, \bar{w}_N\}$ and the parameterization $\pi^{(i)}\big(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)}\big)$, where $\bar{k}_1 = b$
- Instead of treating $(\bar{\mathbf{k}}, \bar{\mathbf{w}})$ as today's states and inferring k' forward, treat them as tomorrow's states (so $k'_{m'} = \bar{k}_{m'}$) and infer current k backward via the Euler equation:

$$\tilde{c}^{(i+1)}_{m',n} = u_c^{-1}\left(\beta R \sum_{n'=1}^{N} P_{nn'}\, u_c\Big(\pi^{(i)}(\bar{k}_{m'},\, \bar{w}_{n'};\, \bar{\mathbf{k}},\, \bar{\mathbf{w}},\, \mathbf{c}^{(i)})\Big)\right).$$

  The right-hand side is explicit (no root finding) because all inputs are known.
- We do not apply the borrowing cap here since $\bar{\pi}(k, w) = Rk + w - b$ depends on the current k, which is not yet known; this will be enforced after we back out $\tilde{k}$ and interpolate onto the current-k grid.
- Here $\tilde{c}^{(i+1)}_{m',n}$ is the unconstrained consumption at off-grid point $\tilde{k}_{m',n}$; it is not yet the desired on-grid value $c^{(i+1)}_{m,n}$.
- Recover the associated current state from the budget constraint:

$$\bar{k}_{m'} = R\,\tilde{k}^{(i+1)}_{m',n} + \bar{w}_n - \tilde{c}^{(i+1)}_{m',n} \quad\Longrightarrow\quad \tilde{k}^{(i+1)}_{m',n} = \frac{\bar{k}_{m'} - \bar{w}_n + \tilde{c}^{(i+1)}_{m',n}}{R}.$$

EGM: Interpolation

- Recall that the target parameterization is $\pi^{(i+1)}\big(k, w;\, \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i+1)}\big)$, while the current (off-grid) parameterization is $\pi^{(i+1)}\big(k, w;\, \tilde{\mathbf{k}}^{(i+1)}, \bar{\mathbf{w}}, \tilde{\mathbf{c}}^{(i+1)}\big)$.
- We apply the interpolation scheme described earlier: for each grid point $\bar{w}_n$ (outer loop) and each grid point $\bar{k}_m$ (inner loop):
  - find adjacent points in $\tilde{\mathbf{k}}^{(i+1)}$ such that $\tilde{k}^{(i+1)}_{m',n} \leqslant \bar{k}_m \leqslant \tilde{k}^{(i+1)}_{m'+1,n}$, then compute the weight

$$\lambda(\bar{k}_m) := \frac{\bar{k}_m - \tilde{k}^{(i+1)}_{m',n}}{\tilde{k}^{(i+1)}_{m'+1,n} - \tilde{k}^{(i+1)}_{m',n}} \in [0, 1]$$

    and therefore

$$\pi^{(i+1)}(\bar{k}_m, \bar{w}_n) := c^{(i+1)}_{m,n} = (1 - \lambda(\bar{k}_m))\, \tilde{c}^{(i+1)}_{m',n} + \lambda(\bar{k}_m)\, \tilde{c}^{(i+1)}_{m'+1,n}.$$

  - if $\bar{k}_m < \tilde{k}^{(i+1)}_{1,n}$: $\pi^{(i+1)}(\bar{k}_m, \bar{w}_n) = \bar{\pi}_{m,n}$ where $\bar{\pi}_{m,n} := R\,\bar{k}_m + \bar{w}_n - b$
  - if $\bar{k}_m > \tilde{k}^{(i+1)}_{M',n}$: use monotone linear extrapolation from $(\tilde{k}^{(i+1)}_{M'-1,n}, \tilde{c}^{(i+1)}_{M'-1,n})$ and $(\tilde{k}^{(i+1)}_{M',n}, \tilde{c}^{(i+1)}_{M',n})$, or clamp: set $\pi^{(i+1)}(\bar{k}_m, \bar{w}_n) = \tilde{c}^{(i+1)}_{M',n}$

Time Iteration (TI): Pseudocode (1/2)

1. Prerequisites:
   - Utility $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$, marginal utility $u_c(c) = c^{-\sigma}$.
   - Parameters β, R, σ, b; transition matrix $P = (P_{nn'})$.
   - Grids: $\bar{\mathbf{k}} = \{\bar{k}_1, \ldots, \bar{k}_M\}$ with $\bar{k}_1 = b$, and $\bar{\mathbf{w}} = \{\bar{w}_1, \ldots, \bar{w}_N\}$.
   - Solver hyperparameters: MaxIter, tolerance ε.
2. Initialize:
   - Borrowing cap (elementwise) $\bar{\pi}_{m,n} := R\,\bar{k}_m + \bar{w}_n - b$.
   - Set $c^{(0)}_{m,n} = \bar{\pi}_{m,n}$ for all (m, n). This defines $\pi^{(0)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(0)})$.
3. Iterate for $i = 0, 1, \ldots, \text{MaxIter} - 1$:
   - For each node (m, n), define the residual and an Nonlinear Complementarity Problem (NCP) reformulation:

$$G_{m,n}(c) := u_c(c) - \beta R \sum_{n'=1}^{N} P_{nn'}\, u_c\Big(\pi^{(i)}(R\bar{k}_m + \bar{w}_n - c,\ \bar{w}_{n'};\ \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)})\Big),$$

$$\Psi_{m,n}(c) := \sqrt{G_{m,n}(c)^2 + (\bar{\pi}_{m,n} - c)^2} - G_{m,n}(c) - (\bar{\pi}_{m,n} - c).$$

Time Iteration (TI): Pseudocode (2/2)

3. Iterate (continued):
   - Solve for $c_{m,n}^{(i+1)} \in [0, \bar{\pi}_{m,n}]$ using either:
     - Bracketing (bisection/Brent) on $G_{m,n}(c) = 0$ with clipping at $\bar{\pi}_{m,n}$, or
     - Semismooth Newton on the Fischer–Burmeister equation $\Psi_{m,n}(c) = 0$.
   - Set $k'_{m,n} = R\,\bar{k}_m + \bar{w}_n - c_{m,n}^{(i+1)}$.
4. Convergence:
   - If $\max\limits_{m,n} \left| c_{m,n}^{(i+1)} - c_{m,n}^{(i)} \right| < \varepsilon$, stop.
   - Otherwise, form $\pi^{(i+1)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i+1)})$ and continue.

Euler Residual Function

- Define the node-wise Euler residual (with $k' = R\bar{k}_m + \bar{w}_n - c$):

$$G_{m,n}(c) := u_c(c) - \beta R \sum_{n'=1}^{N} P_{nn'}\, u_c\Big(\pi^{(i)}(k', \bar{w}_{n'}; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)})\Big).$$

- Monotonicity (unique root): since $u_{cc} < 0$ and $k' = R\bar{k}_m + \bar{w}_n - c$,

$$\frac{dG_{m,n}}{dc} = u_{cc}(c) + \beta R \sum_{n'} P_{nn'}\, u_{cc}\Big(\pi^{(i)}(k', \bar{w}_{n'})\Big)\, \frac{\partial\, \pi^{(i)}(k', \bar{w}_{n'})}{\partial k'} \;<\; 0,$$

  where $\partial \pi^{(i)}/\partial k' \geqslant 0$ by monotonicity in resources, and $u_{cc} < 0$. Hence $G_{m,n}$ is strictly decreasing $\Rightarrow$ at most one root.

- Let $c^\star$ be the unconstrained root, $G_{m,n}(c^\star) = 0$. The feasible set is $[0, \bar{\pi}_{m,n}]$ with $\bar{\pi}_{m,n} := R\bar{k}_m + \bar{w}_n - b$.
  - If $c^\star \leqslant \bar{\pi}_{m,n}$: interior solution $c_{m,n}^{(i+1)} = c^\star$.
  - If $c^\star > \bar{\pi}_{m,n}$: boundary solution $c_{m,n}^{(i+1)} = \bar{\pi}_{m,n}$.

Complementarity (NCP) formulations

- KKT at node (m,n) (an NCP in c):

$$G_{m,n}\big(c_{m,n}^{(i+1)}\big) \;\geqslant\; 0 \;\perp\; \bar{\pi}_{m,n} - c_{m,n}^{(i+1)} \;\geqslant\; 0.$$

- Use the $\min$ operator to unify the interior and binding cases (single-equation NCP):

$$\min\{\, G_{m,n}(c),\; \bar{\pi}_{m,n} - c \,\} = 0.$$

- Fischer–Burmeister (FB) reformulation (single equation, differentiable almost everywhere):

$$\Psi_{m,n}(c) := \sqrt{G_{m,n}(c)^2 + (\bar{\pi}_{m,n} - c)^2} - G_{m,n}(c) - \big(\bar{\pi}_{m,n} - c\big) = 0.$$

- Practical note: the FB map works well with autodiff when we later introduce machine-learning-based solvers.

Endogenous Grid Method (EGM): Pseudocode (1/2)

1. Prerequisites:
   - Utility $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$, marginal utility $u_c(c) = c^{-\sigma}$.
   - Parameters β, R, σ, b; transition matrix $P = (P_{nn'})$.
   - Grids: $\bar{\mathbf{k}} = \{\bar{k}_1, \ldots, \bar{k}_M\}$ with $\bar{k}_1 = b$, and $\bar{\mathbf{w}} = \{\bar{w}_1, \ldots, \bar{w}_N\}$.
   - Solver hyperparameters: MaxIter, tolerance ε.
2. Initialize:
   - Borrowing cap (elementwise) $\bar{\pi}_{m,n} := R\,\bar{k}_m + \bar{w}_n - b$.
   - Set $c^{(0)}_{m,n} = \bar{\pi}_{m,n}$ for all (m, n). This defines $\pi^{(0)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(0)})$.
3. Iterate for $i = 0, 1, \ldots, \text{MaxIter} - 1$:
   - Backward consumption for $\tilde{\mathbf{c}}^{(i+1)}$. For each (m', n):

$$\tilde{c}^{(i+1)}_{m',n} = u_c^{-1}\left(\beta R \sum_{n'=1}^{N} P_{nn'}\, u_c\left(\pi^{(i)}(\bar{k}_{m'},\, \bar{w}_{n'};\, \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)})\right)\right).$$

   - Implied current state $\tilde{\mathbf{k}}$. For each (m', n): $\tilde{k}^{(i+1)}_{m',n} = \frac{\bar{k}_{m'} - \bar{w}_n + \tilde{c}^{(i+1)}_{m',n}}{R}$.

Endogenous Grid Method (EGM): Pseudocode (2/2)

3. Iterate (continued):
   - Interpolation. For each (m, n):
     - If $\bar{k}_m < \tilde{k}_{1,n}^{(i+1)}$: set $c_{m,n}^{(i+1)} = \bar{\pi}_{m,n}$
     - Else if $\bar{k}_m > \tilde{k}_{M,n}^{(i+1)}$: use monotone linear extrapolation from the last two pairs $(\tilde{k}_{M-1,n}^{(i+1)}, \tilde{c}_{M-1,n}^{(i+1)})$, $(\tilde{k}_{M,n}^{(i+1)}, \tilde{c}_{M,n}^{(i+1)})$, or clamp to $c_{m,n}^{(i+1)} = \tilde{c}_{M,n}^{(i+1)}$.
     - Else (interior): find m' s.t. $\tilde{k}_{m',n}^{(i+1)} \leqslant \bar{k}_m \leqslant \tilde{k}_{m'+1,n}^{(i+1)}$, set

$$\lambda(\bar{k}_m) = \frac{\bar{k}_m - \tilde{k}_{m',n}^{(i+1)}}{\tilde{k}_{m'+1,n}^{(i+1)} - \tilde{k}_{m',n}^{(i+1)}}, \quad c_{m,n}^{(i+1)} = (1-\lambda)\,\tilde{c}_{m',n}^{(i+1)} + \lambda\,\tilde{c}_{m'+1,n}^{(i+1)}.$$

     - (Optional) Enforce cap: $c_{m,n}^{(i+1)} \leftarrow \min\{c_{m,n}^{(i+1)}, \bar{\pi}_{m,n}\}$.
   - (Optional) Damping: $c^{(i+1)} \leftarrow (1-\alpha)c^{(i)} + \alpha c^{(i+1)}$, with $\alpha \in (0, 1]$.
   - set $k'_{m,n} = R\,\bar{k}_m + \bar{w}_n - c_{m,n}^{(i+1)}$
4. Convergence:
   - If $\max_{m,n} \left| c_{m,n}^{(i+1)} - c_{m,n}^{(i)} \right| < \varepsilon$, stop.
   - Otherwise, form $\pi^{(i+1)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i+1)})$ and continue.

Forward rollout (lagged-Coleman explicit update): Pseudocode (1/2)

1. Prerequisites:
   - Utility $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$, marginal utility $u_c(c) = c^{-\sigma}$.
   - Parameters β, R, σ, b; transition matrix $P = (P_{nn'})$.
   - Grids: $\bar{\mathbf{k}} = \{\bar{k}_1, \dots, \bar{k}_M\}$ with $\bar{k}_1 = b$, and $\bar{\mathbf{w}} = \{\bar{w}_1, \dots, \bar{w}_N\}$.
   - Solver hyperparameters: MaxIter, tolerance ε.
2. Initialize:
   - Borrowing cap (elementwise) $\bar{\pi}_{m,n} := R\,\bar{k}_m + \bar{w}_n - b$.
   - Set $c^{(0)}_{m,n} = \bar{\pi}_{m,n}$ for all (m, n). This defines $\pi^{(0)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(0)})$.

Forward rollout (lagged-Coleman explicit update): Pseudocode (2/2)

3. Iterate for $i = 0, 1, \dots, \mathsf{MaxIter} - 1$:
   - For each node (m, n), form the next state using the old policy (explicit rollout):
   $$k'^{(i)}_{m,n} \;:=\; R\,\bar{k}_m + \bar{w}_n - c^{(i)}_{m,n}.$$
   - Compute the unconstrained updated consumption via the Euler RHS under $\pi^{(i)}$:
   $$\tilde{c}^{(i+1)}_{m,n} \;=\; u_c^{-1}\!\left(\beta R \sum_{n'=1}^{N} P_{nn'}\; u_c\!\left(\pi^{(i)}\big(k'^{(i)}_{m,n},\; \bar{w}_{n'};\, \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i)}\big)\right)\right).$$
   - Enforce feasibility on today's grid (borrowing cap):
   $$c^{(i+1)}_{m,n} \;=\; \min\big\{\tilde{c}^{(i+1)}_{m,n},\; \bar{\pi}_{m,n}\big\}.$$
   - (Optional) Damping: $c^{(i+1)} \leftarrow (1-\alpha)c^{(i)} + \alpha\, c^{(i+1)}, \quad \alpha \in (0, 1]$.
   - Set $k'_{m,n} = R\,\bar{k}_m + \bar{w}_n - c^{(i+1)}_{m,n}$
4. Convergence:
   - If $\max_{m,n} \big|c^{(i+1)}_{m,n} - c^{(i)}_{m,n}\big| < \varepsilon$, stop.
   - Otherwise, form $\pi^{(i+1)}(k, w; \bar{\mathbf{k}}, \bar{\mathbf{w}}, \mathbf{c}^{(i+1)})$ (your interpolation rule) and continue.

Forward rollout vs TI and EGM

- Versus TI: TI is implicit (solve a root/NCP at each node with $k' = R\bar{k}_m + \bar{w}_n - c_{m,n}^{(i+1)}$). Forward rollout is explicit (uses $k'^{(i)}$ from $c^{(i)}$). Same fixed point if it converges; usually needs damping.
- Versus EGM: EGM fixes a k' grid, inverts Euler to current k, interpolates to the k-grid, enforces the cap on the lower tail. Forward rollout stays on the current $(\bar{k}, \bar{w})$ grid; no back-mapping stage.
- Pros: cheaper per iteration; nodewise decoupled and parallel; only interpolate $\pi^{(i)}(k', \bar{w}_{n'})$ in k'.
- Cons: may oscillate/diverge when $\beta R \approx 1$, with flatter u_c (small σ; large intertemporal elasticity of substitution), or near kinks., or near kinks; Euler holds only in the limit; always re-impose $c_{m,n}^{(i+1)} \leqslant \bar{\pi}_{m,n}$.