# Solution Methods for Dynamic Programming in HA Models

John Ryan

john.p.ryan@wisc.edu

December 30, 2025

# Model setup: Aiyagari

- Standard incomplete markets setup
- Household: preferences $u(c)$, discount factor $\beta$
- HH state is capital holdings $k$, labor productivity shock $\epsilon$ (Markov with $M$)
- Supply 1 unit of labor inelastically, idiosyncratic shock $\epsilon$ uninsurable
- Budget constraint: $c + k' = W\epsilon + Rk$, with $k' \geq \underline{k}$
- Bellman:

$$V(k, \epsilon) = \max_{c, k'} \quad u(c) + \beta \mathbb{E}\left[V(k', \epsilon')|\epsilon\right]$$

- Representative competitive firm: $F(K, L) = K^{\alpha} L^{1-\alpha}$
- Aggregate labor supply $L$ is fixed by law of large numbers
- Final goods: $C + K' = F(K, L) + (1 - \delta)K$

# Steady State Solution Algorithm

1. **Initialize aggregates:** pick $K^{(0)}$; set tolerance $\varepsilon$ and damping parameter $\lambda \in (0, 1]$.
2. **Prices (firm FOCs):**
$$R^{(n)} = \alpha\left(\frac{K^{(n)}}{L}\right)^{\alpha-1} + (1-\delta), \quad W^{(n)} = (1-\alpha)\left(\frac{K^{(n)}}{L}\right)^{\alpha}$$
3. **Solve Household problem backwards (VFI, EGM):**
$$V(k,\epsilon) = \max_{c,k' \geq \underline{k}} u(c) + \beta\,\mathbb{E}[V(k',\epsilon')|\epsilon], \quad \text{yields } g_k^{(n)}(k,\epsilon), V^{(n)}(k,\epsilon)$$
4. **Solve distribution forward (histogram, Monte Carlo):**
$$\mu^{(n)} = T^*(g_k^{(n)}, M)\mu^{(n-1)}$$
5. **Aggregation:**
$$\tilde{K} = \sum_{\epsilon} \int g_k^{(n)}(k,\epsilon)\, d\mu^{(n)}(k,\epsilon)$$
6. **Update aggregates:** if $|\tilde{K} - K^{(n)}| < \varepsilon$ stop; else
$$K^{(n+1)} = (1-\lambda)K^{(n)} + \lambda\,\tilde{K}, \quad \text{and return to Step 2.}$$

# Three Methods Overview

- **Grid Search (VFI)**: discretize choice set $k' \in k_{grid}$. Robust and simple, but slow and coarse.
- **Fitted VFI**: interpolate $V$ or $EV$ across $k$; fewer grid points for similar accuracy, but bottleneck and potential for irregularities with numerical optimization
- **Endogenous Grid Method**: invert Euler to construct an endogenous $k$-grid; typically fastest and lowest error, but watch constraints
- Trade-off between robustness / versatility and efficiency / accuracy
- Note that continuous choices (off grid) necessitates changes to the $T^*$ operator
  - Histogram estimator of Young (2010)

# Grid Search Algorithm

- Discretize choice set and compute the objective value at each grid point
  - The slowest, but simplest and most stable method - brute force

$$V_{n+1}(k,\epsilon) = \max_{k'}\{u(W\epsilon + Rk - k') + \beta\mathbb{E}\left[V_n(k',\epsilon')|\epsilon\right]\}$$

Algorithm

1. Iteration $n$, take as given $V_n(k,\epsilon)$ defined over $(k_{grid}, \epsilon_{grid})$, and prices $R, W$.
2. Initialize $V_{n+1}$ with a low guess $(-\infty)$
3. For each element $k_i \in k_{grid}$, $\epsilon_m \in \epsilon_{grid}$:
   - For each element $k'_j \in k_{grid}$:
     - Compute objective $v(k'_j; k_i, \epsilon_m) := u(W\epsilon_m + Rk_i - k'_j) + \beta\mathbb{E}[V_n(k'_j, \epsilon')]$
   - $V_{n+1}(k_i, \epsilon_m) := \max_{k'_j \in k_{grid}} v(k'_j; k_i, \epsilon_m)$

# Fitted VFI Algorithm

- Like grid search, but interpolate $V$ and numerically optimize for continuous $k'$

$$V_{n+1}(k,\epsilon) = \max_{k'}\{u(W\epsilon + Rk - k') + \beta\mathbb{E}\left[V_n(k',\epsilon')|\epsilon\right]\}$$

Algorithm

1. Iteration $n$, take as given $V_n(k,\epsilon)$ defined over $k_{grid}$, and prices $R, W$.
2. Interpolate[1] $V_n(k,\epsilon)$ across $k_{grid}$ for each value of $\epsilon$: $\tilde{V}_\epsilon(k)$
3. For each element $k_i \in k_{grid}$, $\epsilon_m \in \epsilon_{grid}$:
   - Define continuous objective function: $v(k'; k_i, \epsilon_m) := u(W\epsilon_m + Rk_i - k') + \beta\mathbb{E}[\tilde{V}_{\epsilon'}(k')]$
   - Taking weighted averages of interpolated values $\tilde{V}_{\epsilon'}(k')$
   - Numerically optimize the objective function $v(k'; k_i, \epsilon_m)$
   - Store maximum as $V_{n+1}(k_i, \epsilon_m)$ and argmax is $g_k(k_i, \epsilon_m)$

[1] The choice of interpolator is not innocuous. This is discussed further later.

# Why is this better?

- Fitted VFI requires less computation per iteration and reduces discretization error
- Suppose we have $N_k$ elements of $k_{grid}$
- For each point $(k, \epsilon)$, grid search requires $N_k$ computations of the objective[2]
- Using an optimizer like Brent, fitted VFI can optimize in as few as 5-10
- Further, reduced Euler error from the smooth interpolation admits a coarser grid
- Interpolation is typically a cheap operation and overhead from numerical optimization is small in comparison to gains
- But we can do even better with EGM

[2] This can be improved by exploiting monotonicity in 2 ways: first that $g_k(k_i, \epsilon_m) \leq g_k(k_{i+1}, \epsilon_m)$, and second that $k'_j$ unaffordable $\implies$ $k'_{j+1}$ unaffordable

# Endogenous Grid Method (EGM)

- So far, we haven't exploited the structure of the problem much
- Both previous methods substitute $c$ from BC into objective and optimize
- Can improve our solution by using optimality conditions
- Build Lagrangian:

$$\mathcal{L} = u(c) + \beta \mathbb{E}[V(k', \epsilon')] + \lambda(W\epsilon + Rk - c - k') + \gamma(k' - \underline{k})$$

- Combining FOCs, we get $u'(c) \geq \beta \ \mathbb{E}[\partial_k V(k', \epsilon')]$
  - Holds with equality when borrowing constraint not binding
- We also have the envelope condition: $\partial_k V_t(k, \epsilon) = R \cdot u'(c_t^*(k, \epsilon))$
- Combining, we get the Euler equation: $u'(c^*(k, \epsilon)) = \beta R \ \mathbb{E}[u'(c^*(k', \epsilon'))]$
- Inverting,

$$c^*(k, \epsilon) = (u')^{-1} \left[\beta R \ \mathbb{E}[u'(c^*(k', \epsilon'))]\right]$$

# EGM Algorithm

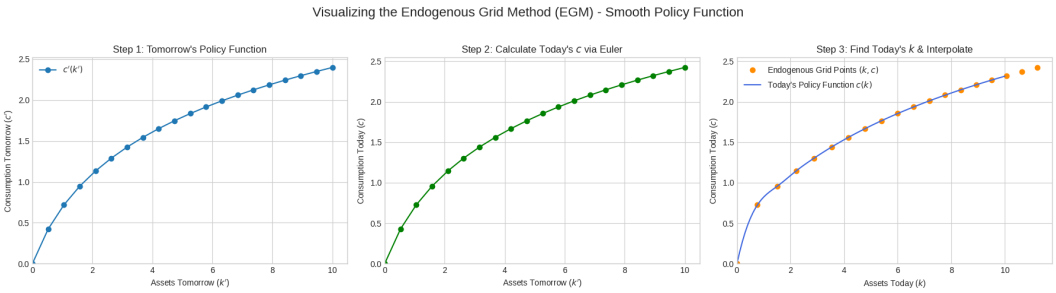
- Problem: $k'$ is an endogenous choice
  - Solution: treat $k'$ as known and solve for $k$ from budget constraint instead
  - Then, interpolate to get back on grid, and check borrowing constraint ex post

$$c^*_{n+1}(k,\epsilon) = (u')^{-1}\left[\beta R\ \mathbb{E}[u'(c^*_n(k',\epsilon'))]\right]$$

Algorithm

1. Take as given current iteration consumption policy function $c^*_n(k',\epsilon')$
2. For each choice $k'_i \in k_{grid}$, productivity $\epsilon_m$:
   - Calculate consumption today $\hat{c}(k'_i,\epsilon_m)$ from Euler equation
   - Infer capital holdings today from BC: $\hat{k}(k'_i,\epsilon_m) = (\hat{c}(k'_i,\epsilon_m) + k'_i - W\epsilon_m)/R$
3. Interpolate across endogenous grid $\hat{k}$ for each $\epsilon_m$ to get policies back on grid: $c^*_{n+1}(k,\epsilon) = ITP(\hat{k},\hat{c})$, $g_{n+1} = ITP(\hat{k}, k'_{grid})$
4. Wherever $g_{n+1}(k,\epsilon) < \underline{k}$, correct with $g_{n+1}(k,\epsilon) = \underline{k}$, and $c^*_{n+1}(k,\epsilon) = Rk + W\epsilon - \underline{k}$

# EGM Visualized



# VFI vs. EGM

Typical VFI algorithm

- Take HH state $(k_t, \epsilon_t)$ and value function $V_{t+1}$ as given
- Solve for $k_{t+1}$ by maximizing Bellman operator
- Solve for $c_t$ from budget constraint
- Borrowing constraint guaranteed satisfied ($\min k_{grid} = \underline{k}$)

EGM flips this on its head:

- Take choice as given (state tomorrow) $k_{t+1}$ and productivity today $\epsilon_t$
- Solve for consumption today $c_t$ from Euler equation
- Solve for $k_t$ from budget constraint

# EGM Recap

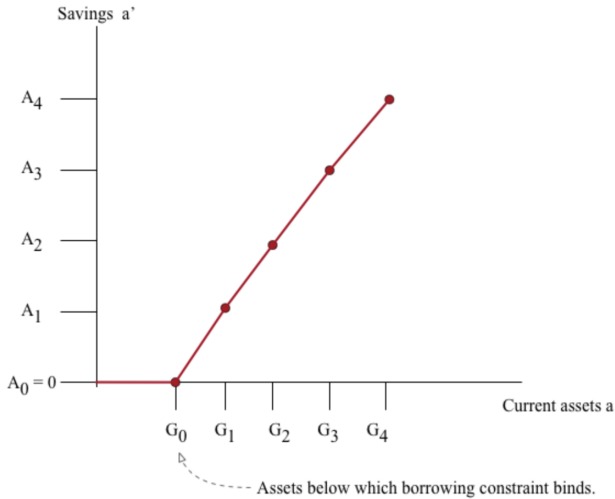
Inverted Euler equation:

$$u'(c_t^*(k_t, \epsilon_t)) \geq \beta R \; \mathbb{E}[u'(c_{t+1}^*(k_{t+1}, \epsilon'))]$$

EGM Steps:

- Take next period's policy function as given $c_{t+1}^*(k_{t+1}, \epsilon_{t+1})$
- Take as given capital tomorrow $k_{t+1} \in k_{grid}$, productivity today $\epsilon_t$
- Find consumption today $c_t$ from inverted Euler equation
- Get capital today from budget constraint: $k_t = \frac{1}{R}[W\epsilon_t - k_{t+1} - c_t]$
- Interpolate across $(k_t, c_t)$ and $(k_t, k_{t+1})$ to get policy functions back on grid
- Check for where borrowing constraint is binding and set $k_{t+1} := \underline{k}$, $c_t = W\epsilon_t + Rk_t - \underline{k}$

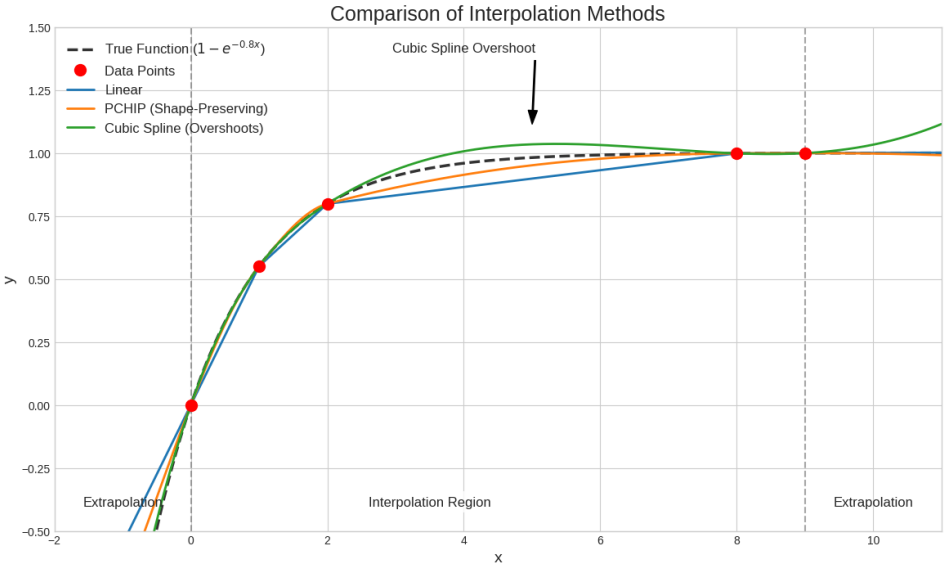
# EGM Visualized



# Interpolation Best Practices

- Need to be careful about preserving concavity and monotonicity
- Consider a linear extrapolation of $V(k, \epsilon)$
  - There might not be an interior solution for the household problem
  - Solution could diverge across iterations
- Consider non-monotone interpolation of $V(k, \epsilon)$
  - Numerical solver can get caught on local optima
  - Non-invertible policy functions lead to failure of EGM
- Linear interpolation preserves monotonicity but not concavity
- Cubic spline can represent shape well but not monotone
- Shape preserving cubic spline (PCHIP) is often a good middle ground (limited in Julia)

# Interpolation Comparison



# Histogram Estimate for $\mu$

- The $T^*$ operator to update $\mu$ given $g(k, \epsilon)$ is simple with grid search
  - Basically a matrix of indicator functions for matching $g(k, \epsilon) \in k_{grid}$
- When policy functions continuous, $g(k, \epsilon) \notin k_{grid}$
- Solution: split the mass between neighboring grid points based on distance (Eric Young's method)
- This is like a linear interpolation of the pdf of agents
- Actually becomes more accurate
- Alternative: Monte Carlo simulation
  - This is slower and less accurate due to sampling variance

# Histogram Estimate for $\mu$

- Case 1: $k_j < g(k_i, \epsilon_m) < k_{j+1}$:

$$\mu'(k_{j+1}, \epsilon') = \sum_{k_i, \epsilon_m} M(\epsilon'|\epsilon_m) \left[ \frac{k_{j+1} - g(k_i, \epsilon_m)}{k_{j+1} - k_j} \right] \mu(k_i, \epsilon_m)$$

$$\mu'(k_j, \epsilon') = \sum_{k_i, \epsilon_m} M(\epsilon'|\epsilon_m) \left[ \frac{g(k_i, \epsilon_m) - k_j}{k_{j+1} - k_j} \right] \mu(k_i, \epsilon_m)$$

- Case 2: $g(k_i, \epsilon_m)$ falls outside $k_{grid}$
  - Attribute all the mass to nearest grid point ($\underline{k}$ or $\max\{k_{grid}\}$)

# EGM Conesa & Krueger

- Endogenous labor supply, nonseparable preferences
- Worker utility $u(c,\ell) = \frac{[c^\gamma(1-\ell)^{1-\gamma}]^{1-\sigma}}{1-\sigma}$
- MU consumption: $u_c(c,\ell) = \frac{\gamma(c^\gamma(1-\ell)^{1-\gamma})^{1-\sigma}}{c}$
- Consumption - labor optimality condition: $\frac{u_\ell(c,\ell)}{u_c(c,\ell)} = -\frac{1-\gamma}{\gamma}\frac{c}{1-\ell} = -w(1-\theta)e(z,\eta_j)$
- Solve for $1-\ell$, can get $u'(c)$ without $\ell$
- Euler equation:
$$\kappa(j,z)c_j^{-\sigma} = \beta(1+r)\mathbb{E}[\kappa(j+1,z')c_{j+1}^{-\sigma}]$$
- Where $\kappa(j,z) = \gamma(\frac{1-\gamma}{\gamma w(1-\theta)e(z,\eta_j)})^{(1-\gamma)(1-\sigma)}$