Quantitative Macroeconomics w/ AI and ML
Lec. 8: Advanced Numerical Methods and Accuracy Assessment

Zhigang Feng

Dec., 2025

Advanced Numerical Methods and Accuracy Assessment

Motivation

- Standard VFI and time iteration are reliable but can be slow.
- Sources of computational cost:
  - Many iterations needed (especially when β close to 1)
  - Maximization or root-finding at each grid point per iteration
  - Fine grids required for accuracy
- This lecture: techniques to accelerate convergence and avoid costly operations.
- Also: rigorous framework for assessing accuracy of numerical solutions.

The Model Environment

- We work with the deterministic optimal growth model:

$$\max_{\{c_t, k_{t+1}\}_{t\geq 0}} \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t. } c_t + k_{t+1} = f(k_t) + (1-\delta)k_t$$

- $f(k) = Ak^\alpha$: production function
- δ: depreciation rate
- State variable: k_t (capital)
- Control: c_t (consumption) or equivalently k_{t+1}
- Policy function: $k' = g(k)$
- For stochastic extensions with productivity z_t:

$$\max \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t. } c_t + k_{t+1} = z_t f(k_t) + (1-\delta)k_t$$

Euler Equation and Optimality

- The Euler equation characterizes optimal consumption:

$$u'(c_t) = \beta u'(c_{t+1}) \left[f'(k_{t+1}) + 1 - \delta \right]$$

- Define gross return: $R(k') = f'(k') + 1 - \delta = \alpha A (k')^{\alpha - 1} + 1 - \delta$
- In terms of policy function $g(k)$:

$$u'(f(k) + (1-\delta)k - g(k)) = \beta u'(f(g(k)) + (1-\delta)g(k) - g(g(k)))R(g(k))$$

- This is a functional equation in $g(\cdot)$.
- All methods (VFI, time iteration, EGM) seek g^* satisfying this condition.

Numerical Representation

- Capital grid: $\mathbb{K} = \{k_1, k_2, \ldots, k_M\}$ with $k_1 = \underline{k}$, $k_M = \bar{k}$
- Policy represented as array: $g_m = g(k_m)$ for $m = 1, \ldots, M$
- Resource constraint at grid point k_m:

$$c_m = f(k_m) + (1 - \delta)k_m - g_m$$

- Interpolation extends policy to continuous domain:

$$g(k; \mathbb{K}, \{g_m\}) \quad \text{(linear, cubic spline, etc.)}$$

- Goal: find $\{g_m^*\}$ such that Euler equation holds at all grid points.

Recap: Standard VFI Algorithm

- **Initialize:** Value function $V^{(0)}(k_m)$ for all m (e.g., $V^{(0)} = 0$).
- **Iterate:** For $n = 0, 1, 2, \ldots$
  1. At each grid point k_m, solve:

$$V^{(n+1)}(k_m) = \max_{k' \in \mathbb{K}} \left[u(f(k_m) + (1-\delta)k_m - k') + \beta V^{(n)}(k') \right]$$

  2. Record optimal choice: $g^{(n+1)}(k_m) = \arg\max$
- **Stop:** When $\|V^{(n+1)} - V^{(n)}\|_\infty < \text{tol}$.
- **Cost per iteration:** $O(M^2)$ operations (search over M choices at M points).
- **Number of iterations:** $O\left(\frac{\log(\text{tol})}{\log(\beta)}\right)$ — slow when $\beta \approx 1$.

Standard VFI: Computational Bottlenecks

- Two main sources of computational cost:
  1. **Many iterations:** Convergence rate β^n is slow for $\beta = 0.96$–0.99.
  2. **Costly per iteration:** Full maximization at every grid point, every iteration.
- Example: $\beta = 0.96$, tolerance 10^{-6}

$$n \geq \frac{\log(10^{-6})}{\log(0.96)} \approx 339 \text{ iterations}$$

- With $M = 500$ grid points: $339 \times 500^2 \approx 85$ million comparisons.
- Acceleration techniques target both bottlenecks.

Acceleration Techniques

Acceleration: Overview

- Three main approaches to speed up convergence:
  1. **Howard's policy improvement**: reduce iterations by exploiting policy stability
  2. **MacQueen-Porteus bounds**: tighter error bounds for early termination
  3. **Exploiting structure**: use monotonicity and concavity to speed up optimization
- These techniques can be combined.
- Potential speedups: 10x–100x over naive VFI.

Howard's Policy Improvement Algorithm

- Key observation: policy function often converges faster than value function.
- Idea: hold policy fixed, iterate on value function (cheap), then update policy.
- Standard VFI: maximize at every iteration.
- Howard's improvement: maximize only every H iterations.
- Between policy updates, compute value given fixed policy:

$$V^{(n+1)}(k_m) = u(f(k_m) + (1-\delta)k_m - g^{(n)}(k_m)) + \beta V^{(n)}(g^{(n)}(k_m))$$

- No maximization $\Rightarrow O(M)$ instead of $O(M^2)$ per iteration.

Howard's Algorithm: Implementation

1. **Initialize:** $V^{(0)}$, policy $g^{(0)}$, parameter H (e.g., $H = 20$).
2. **Policy evaluation** (repeat H times):

$$V^{(n+1)}(k_m) = u(c_m) + \beta V^{(n)}(g^{(n)}(k_m))$$

   where $c_m = f(k_m) + (1-\delta)k_m - g^{(n)}(k_m)$.
3. **Policy improvement:**

$$g^{(n+1)}(k_m) = \arg\max_{k' \in \mathbb{K}} \left[u(f(k_m) + (1-\delta)k_m - k') + \beta V^{(n+H)}(k') \right]$$

4. **Check convergence:** If $\|g^{(n+1)} - g^{(n)}\|_\infty <$ tol, stop.
5. Return to step 2.

Note: Here $k' \in \mathbb{K}$ restricts choices to grid points for simplicity. With interpolation, k' can be continuous and we use $V^{(n)}(k'; \mathbb{K}, \{V_m\})$.

Howard's Algorithm: Why It Works

- Policy evaluation is a linear fixed point problem.
- Given policy g, the value function satisfies:

$$V_g = u_g + \beta P_g V_g \quad \Rightarrow \quad V_g = (I - \beta P_g)^{-1} u_g$$

  where P_g is the transition matrix induced by policy g.
- Can solve exactly (matrix inversion) or iterate (fast convergence).
- Policy improvement guarantees $V^{g^{(n+1)}} \geq V^{g^{(n)}}$ pointwise.
- Finite number of possible policies on discrete grid $\Rightarrow$ terminates in finite steps.

Howard's Algorithm: Performance

- Typical results for growth model ($\beta = 0.96$, $M = 500$):

Method	Policy Updates	Time (relative)
Standard VFI	1,500+	1.0
Howard ($H = 10$)	150	0.15
Howard ($H = 20$)	80	0.10
Howard ($H = 50$)	40	0.08

- Diminishing returns: very large H adds evaluation cost without much benefit.
- Rule of thumb: $H \in [10, 50]$ works well for most problems.

MacQueen-Porteus Bounds: Motivation

- Standard stopping criterion: $\|V^{(n+1)} - V^{(n)}\|_\infty < \text{tol}$
- Problem: this bounds change between iterates, not distance to true V^*.
- We want to know: $\|V^{(n)} - V^*\|_\infty \leq ?$
- MacQueen (1966), Porteus (1971): exploit contraction structure to get tighter bounds.
- Key insight: if changes are uniform across states, we are close to convergence.

MacQueen-Porteus Bounds: Theory

- After iteration n, compute:

$$d_{\min}^{(n)} = \min_m \left[V^{(n+1)}(k_m) - V^{(n)}(k_m) \right]$$
$$d_{\max}^{(n)} = \max_m \left[V^{(n+1)}(k_m) - V^{(n)}(k_m) \right]$$

- **Theorem:** The true value function V^* satisfies:

$$V^{(n+1)}(k_m) + \frac{\beta}{1-\beta} d_{\min}^{(n)} \leq V^*(k_m) \leq V^{(n+1)}(k_m) + \frac{\beta}{1-\beta} d_{\max}^{(n)}$$

- Error bound:

$$\|V^{(n+1)} - V^*\|_\infty \leq \frac{\beta}{1-\beta} \left(d_{\max}^{(n)} - d_{\min}^{(n)} \right)$$

MacQueen-Porteus Bounds: Implementation

- **Stopping criterion:** Stop when

$$\frac{\beta}{1-\beta}\left(d_{\max}^{(n)} - d_{\min}^{(n)}\right) < \text{tol}$$

- This is tighter than $\|V^{(n+1)} - V^{(n)}\|_\infty < \text{tol}$ in many cases.
- **Acceleration:** Use bounds to "center" the approximation:

$$\tilde{V}^{(n+1)}(k_m) = V^{(n+1)}(k_m) + \frac{\beta}{1-\beta} \cdot \frac{d_{\min}^{(n)} + d_{\max}^{(n)}}{2}$$

- This shifts $V^{(n+1)}$ toward the middle of the error bounds.
- Combines well with Howard's improvement for further speedup.

Exploiting Monotonicity

- Many economic problems have monotone policy functions.
- In growth model: $g(k)$ is increasing in k.
- Standard VFI: search over all $k' \in \mathbb{K}$ at each grid point.
- With monotonicity: if optimal k' at k_m is k_{j^*}, then optimal k' at $k_{m+1} \geq k_{j^*}$.
- Search only over $\{k_{j^*}, k_{j^*+1}, \ldots, k_M\}$ for next grid point.
- Reduces search from $O(M^2)$ to $O(M)$ per iteration.

Exploiting Monotonicity: Implementation

1. Order capital grid: $k_1 < k_2 < \cdots < k_M$.
2. Initialize lower bound index: $j_{\min} = 1$.
3. For $m = 1, \ldots, M$:
   - Search for optimal k' over $\{k_{j_{\min}}, \ldots, k_M\}$ only.
   - Let j^* be the index of optimal choice.
   - Set $g(k_m) = k_{j^*}$.
   - Update: $j_{\min} \leftarrow j^*$.

Speedup: From $O(M^2)$ to $O(M)$ comparisons per iteration.

Exploiting Concavity

- Value function is concave $\Rightarrow$ objective in maximization is concave in k'.
- No need for exhaustive grid search; use efficient algorithms:
  - **Golden section search**: guaranteed convergence, no derivatives needed
  - **Brent's method**: combines bisection with inverse quadratic interpolation
  - **Newton's method**: fast quadratic convergence if derivatives available
- Concavity + monotonicity: combine both speedups.
- With continuous choice and interpolation: $O(\log M)$ per grid point.

Combining Acceleration Techniques

- Best practice: combine multiple techniques.
- Recommended approach:
  1. Use monotonicity to restrict search range
  2. Use concavity for efficient optimization (golden section)
  3. Apply Howard's improvement ($H = 20$)
  4. Use MacQueen-Porteus for early stopping
- Typical combined speedup: 50x–200x over naive VFI.
- Code complexity increases; worth it for repeated solutions (estimation, simulation).

Endogenous Grid Method (EGM)

Time Iteration: The Bottleneck

- Recall time iteration: given $g^{(i)}$, find $g^{(i+1)}$ satisfying Euler equation.
- At each grid point k_m, solve for k':

$$u'(f(k_m) + (1-\delta)k_m - k') = \beta u'(c(k'; g^{(i)}))R(k')$$

  where $c(k'; g^{(i)}) = f(k') + (1-\delta)k' - g^{(i)}(k')$.
- Problem: k' appears on both sides of the equation.
- Requires **root-finding** at each grid point, each iteration.
- Computationally expensive with many grid points.

EGM: The Key Insight

- Standard approach: fix today's state k, solve for tomorrow's k'.
- **EGM insight** (Carroll 2006): fix tomorrow's state k', solve for today's k.
- Treat the capital grid $\mathbb{K}$ as **tomorrow's** capital values.
- For each $k' \in \mathbb{K}$:
  1. Use Euler equation to solve for current consumption c **explicitly**.
  2. Use budget constraint to back out current capital k.
- No root-finding required!

EGM: Step 1 — Solve for Current Consumption

- Fix tomorrow's capital at grid point: $k' = k_{m'}$ for some $m' \in \{1, \ldots, M\}$.
- Given current policy guess $g^{(i)}$, tomorrow's consumption is:

$$c' = f(k_{m'}) + (1 - \delta)k_{m'} - g^{(i)}(k_{m'})$$

- Euler equation:

$$u'(c) = \beta u'(c')R(k_{m'})$$

- Solve explicitly by inverting marginal utility:

$$\tilde{c}_{m'}^{(i+1)} = (u')^{-1}\left(\beta u'(c')R(k_{m'})\right)$$

- For CRRA utility $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$: $(u')^{-1}(x) = x^{-1/\sigma}$

EGM: Step 2 — Recover Current Capital

- Budget constraint: $c + k' = f(k) + (1 - \delta)k$
- With $k' = k_{m'}$ and $c = \tilde{c}_{m'}^{(i+1)}$, solve for k:

$$\tilde{k}_{m'}^{(i+1)} + \tilde{c}_{m'}^{(i+1)} = f(\tilde{k}_{m'}^{(i+1)}) + (1 - \delta)\tilde{k}_{m'}^{(i+1)}$$

- This gives the **endogenous** current capital $\tilde{k}_{m'}^{(i+1)}$.
- Key observation: $k_{m'}$ (tomorrow) is on the grid $\mathbb{K}$, but $\tilde{k}_{m'}^{(i+1)}$ (today) is **not** on the grid.
- We have pairs: $(\tilde{k}_{m'}^{(i+1)}, k_{m'})$ for $m' = 1, \ldots, M$.

EGM: Step 3 — Interpolate to Exogenous Grid

- Have: policy $k' = k_{m'}$ at endogenous points $\tilde{k}_{m'}^{(i+1)}$.
- Want: policy $g^{(i+1)}(k_m)$ at exogenous grid points $k_m \in \mathbb{K}$.
- For each k_m:
  - Find m' such that $\tilde{k}_{m'}^{(i+1)} \leq k_m \leq \tilde{k}_{m'+1}^{(i+1)}$
  - Interpolate:

$$\lambda = \frac{k_m - \tilde{k}_{m'}^{(i+1)}}{\tilde{k}_{m'+1}^{(i+1)} - \tilde{k}_{m'}^{(i+1)}}, \quad g^{(i+1)}(k_m) = (1-\lambda)k_{m'} + \lambda k_{m'+1}$$

- This maps the endogenous grid back to the exogenous grid.

EGM: Visual Illustration

- Standard time iteration:

$$\text{Given } k \in \mathbb{K} \xrightarrow{\text{root-finding}} k' = g(k)$$

- EGM:

$$\text{Given } k' \in \mathbb{K} \xrightarrow{\text{explicit}} (c, k) \xrightarrow{\text{interpolate}} g(k) \text{ on } \mathbb{K}$$

- The "endogenous grid" refers to the computed values $\{\tilde{k}_{m'}\}$ which depend on the current policy guess.
- After interpolation, we have $g^{(i+1)}$ defined on the original exogenous grid $\mathbb{K}$.

EGM: Pseudocode

Initialize: Grid $\mathbb{K} = \{k_1, \ldots, k_M\}$; initial policy $g^{(0)}(k_m) = k_m$ (or steady state).

Iterate for $i = 0, 1, \ldots$:

1. For each $m' = 1, \ldots, M$:
   - $c' = f(k_{m'}) + (1-\delta)k_{m'} - g^{(i)}(k_{m'})$
   - $\tilde{c}_{m'}^{(i+1)} = (u')^{-1}\left(\beta u'(c') R(k_{m'})\right)$
   - Solve for $\tilde{k}_{m'}^{(i+1)}$ from: $\tilde{c}_{m'}^{(i+1)} + k_{m'} = f(\tilde{k}_{m'}^{(i+1)}) + (1-\delta)\tilde{k}_{m'}^{(i+1)}$
2. Interpolate $\{(\tilde{k}_{m'}^{(i+1)}, k_{m'})\}$ to get $g^{(i+1)}(k_m)$ on $\mathbb{K}$.
3. If $\max_m |g^{(i+1)}(k_m) - g^{(i)}(k_m)| < \epsilon$: stop.

EGM: Handling the Budget Constraint Inversion

- Step 2 requires solving for $\tilde{k}$ from:

$$\tilde{c} + k' = f(\tilde{k}) + (1 - \delta)\tilde{k}$$

- Rearranging: $\tilde{k}$ solves $f(\tilde{k}) + (1 - \delta)\tilde{k} - \tilde{c} - k' = 0$
- For Cobb-Douglas $f(k) = Ak^{\alpha}$: this requires numerical root-finding.
- However, this is a **one-dimensional** problem with known structure.
- Alternative: for simpler budget constraints (e.g., $c + k' = Rk + w$), inversion is explicit:

$$\tilde{k} = \frac{k' + \tilde{c} - w}{R}$$

EGM vs. Time Iteration: Comparison

	Time Iteration	EGM
Grid interpretation	Today's states	Tomorrow's states
Euler equation	Implicit in k'	Explicit in c
Main computation	Root-finding	Function inversion
Per-iteration cost	High	Low
Implementation	More complex	Simpler

Typical speedup: EGM is 5x–20x faster than standard time iteration.

EGM: Computational Gains

- Main gain: eliminates root-finding in Euler equation.
- Time iteration: root-finding at each of M grid points per iteration.
- EGM: explicit formula + interpolation.
- Root-finding typically requires 10–50 function evaluations per point.
- Additional benefits:
  - Operations are vectorizable (element-wise utility inversion)
  - Naturally parallelizable across grid points
  - Can leverage modern numerical libraries

EGM: Extensions

- **Stochastic models:** Add expectation over future shocks; EGM still applies.
- **Borrowing constraints:** Handled via interpolation (constrained region identified automatically).
- **Multiple assets:** Partial EGM still beneficial; some dimensions require root-finding.
- **Non-convex problems:** Upper envelope methods (Fella 2014) handle discrete choices.
- **Life-cycle models:** EGM particularly efficient for finite-horizon backward induction.

Method Comparison: Speed vs. Robustness

Method	Speed	Robustness	Complexity
Standard VFI	Slow	High	Low
VFI + Howard	Fast	High	Medium
Time Iteration	Moderate	High	Medium
EGM	Fast	High	Medium

Recommendation:

- Development/debugging: Standard VFI (reliable baseline)
- Production: EGM or VFI + Howard (speed + robustness)
- Complex constraints: Time iteration with complementarity formulation

Convergence and Error Analysis

Theoretical Framework: Assumptions

- Following Santos (1998), assume:
  - State space is compact, feasible set for (k, k') is convex.
  - Return function u is continuous, C^2 with bounded derivatives.
  - Strong concavity: $\exists \eta > 0$ such that $u(c) + \frac{1}{2}\eta c^2$ is concave.
- Implications:
  - Optimal paths are interior (away from boundaries).
  - True value function V^* is C^2.
  - True policy function g^* is C^1.
- These regularity conditions enable sharp error bounds.

Convergence of VFI

- **Lemma (Convergence):** Under stated assumptions, numerical VFI has unique limit $V^h \in \mathcal{W}^h$. This limit can be approached to arbitrary precision in finite iterations.
- Rate of convergence:
$$\|V^{(n)} - V^h\| \leq \beta^n \|V^{(0)} - V^h\|$$
- Number of iterations to achieve tolerance ϵ:
$$n \geq \frac{\log(\epsilon) - \log(\|V^{(0)} - V^h\|)}{\log(\beta)}$$
- Slow convergence when β close to 1 (typical in macro: $\beta \approx 0.96$–0.99).

Error Bounds: Santos (1998)

- **Theorem:** Let V, g be true value and policy functions, and V^h, g^h be numerical solutions on grid with mesh size h. Then:

$$\|V - V^h\| \leq \frac{M}{1-\beta} h^2$$

$$\|g(k_m) - g^h(k_m)\| \leq \left(\frac{M}{1-\beta}\right)^{1/2} h \quad \text{for all grid points } k_m$$

- M is a constant depending on model primitives (bounds on second derivatives).
- Key insight:
  - Value function error: quadratic in mesh size h
  - Policy function error: linear in mesh size h
  - Halving grid spacing $\Rightarrow$ 4x better value, 2x better policy

Intuition for Error Bounds

- Let T, T^h denote exact and discretized Bellman operators.
- Triangle inequality:

$$\|V - V^h\| = \|TV - T^hV^h\| \leq \|TV - T^hV\| + \|T^hV - T^hV^h\|$$

- By contraction property:

$$\|T^hV - T^hV^h\| \leq \beta\|V - V^h\|$$

- Rearranging:

$$\|V - V^h\| \leq \frac{\|TV - T^hV\|}{1 - \beta}$$

- The term $\|TV - T^hV\|$ is the one-step approximation error, bounded by $\gamma h^2/2$ where γ bounds second derivative of V.

Practical Convergence Criteria

- **Sup norm:** $\|V^{(n+1)} - V^{(n)}\|_\infty = \max_m |V^{(n+1)}(k_m) - V^{(n)}(k_m)| < \text{tol}$
  - Most common criterion
  - Directly related to theory
- **Relative difference:** $\max_m \frac{|V^{(n+1)}(k_m) - V^{(n)}(k_m)|}{|V^{(n)}(k_m)|} < \text{tol}$
  - Scale-invariant
  - Better for problems with large value function magnitudes
- **Policy function convergence:** $\|g^{(n+1)} - g^{(n)}\|_\infty < \text{tol}$
  - Often converges faster than value function
  - Directly relevant for simulation
- Typical tolerance: 10^{-6} to 10^{-8}.

Accuracy Assessment

Why Accuracy Assessment Matters

- Numerical solutions are approximations—how good?
- Sources of error:
  - Discretization (finite grid, mesh size h)
  - Interpolation scheme
  - Convergence tolerance
  - Floating-point arithmetic
- Need metrics to:
  - Validate solutions against theoretical bounds
  - Compare methods on equal footing
  - Choose appropriate grid density
  - Report accuracy in publications

The Evaluation Principle: Separation of Grids

- **Key principle:** Evaluate accuracy on a **different grid** than the solution grid.
- Analogy to machine learning:
  - Solution grid $\mathbb{K} = \{k_1, \ldots, k_M\} \leftrightarrow$ Training set
  - Evaluation grid $\mathbb{K}^{\text{test}} = \{k_1^{\text{test}}, \ldots, k_L^{\text{test}}\} \leftrightarrow$ Test set
- Why separation matters:
  - Solution is constructed to satisfy conditions *at grid points*
  - Errors at grid points may be artificially small
  - True test: does solution generalize to off-grid points?
- Evaluation grid should be finer (e.g., $L = 10M$) and not overlap with $\mathbb{K}$.

Euler Equation Errors: Definition

- At the true solution, Euler equation holds exactly:

$$u'(c) = \beta u'(c')R(k')$$

- For approximate policy $\hat{g}$, define residual at **off-grid** point k:

$$\mathcal{E}(k) = u'(\hat{c}) - \beta u'(\hat{c}')R(\hat{g}(k))$$

  where:
  - $\hat{c} = f(k) + (1-\delta)k - \hat{g}(k)$ (interpolated policy)
  - $\hat{c}' = f(\hat{g}(k)) + (1-\delta)\hat{g}(k) - \hat{g}(\hat{g}(k))$
- Residual $\mathcal{E}(k) \neq 0$ measures approximation error at k.
- Connection to theory: small $\mathcal{E}$ implies small policy error (Santos 2000).

Unit-Free Euler Equation Errors

- Raw residual $\mathcal{E}$ depends on units; hard to interpret.
- **Unit-free error** (Judd 1992):

$$\mathsf{EE}(k) = \left| 1 - \frac{c^*}{\hat{c}} \right|$$

  where c^* is the consumption that would exactly satisfy the Euler equation.
- Alternative formulation:

$$\mathsf{EE}(k) = \left| 1 - \frac{\beta u'(\hat{c}') R(\hat{g}(k))}{u'(\hat{c})} \right|$$

- Interpretation: fraction of consumption the agent would adjust if re-optimizing.

Euler Errors: Interpretation and Standards

- $\text{EE}(k) = 10^{-3}$: agent would change consumption by 0.1%.
- $\text{EE}(k) = 10^{-5}$: agent would change consumption by 0.001%.
- Typical reporting standards:

$\log_{10}(\max \text{EE})$	Assessment
> -2	Poor
-3 to -2	Acceptable
-5 to -3	Good
-7 to -5	Very good
< -7	Excellent

- Report both $\max$ and mean across evaluation grid.

Computing Euler Errors: Algorithm

1. **Construct evaluation grid:** $\mathbb{K}^{\text{test}} = \{k_1^{\text{test}}, \ldots, k_L^{\text{test}}\}$
   - Finer than solution grid (e.g., $L = 10M$)
   - Points should **not** coincide with $\mathbb{K}$
2. **For each** $k_j^{\text{test}} \in \mathbb{K}^{\text{test}}$:
   - Interpolate: $\hat{k}' = \hat{g}(k_j^{\text{test}}; \mathbb{K}, \{g_m\})$
   - Compute: $\hat{c} = f(k_j^{\text{test}}) + (1-\delta)k_j^{\text{test}} - \hat{k}'$
   - Interpolate: $\hat{k}'' = \hat{g}(\hat{k}')$
   - Compute: $\hat{c}' = f(\hat{k}') + (1-\delta)\hat{k}' - \hat{k}''$
   - Error: $\text{EE}_j = |1 - \beta u'(\hat{c}')R(\hat{k}')/u'(\hat{c})|$
3. **Report:** $\max_j \text{EE}_j$ and $\text{mean}_j \text{EE}_j$

Euler Errors: Visualization

- Plot $\log_{10}(\text{EE}(k))$ across the evaluation grid.
- Typical patterns:
  - Errors larger near boundaries of state space
  - Errors larger in regions of high curvature
  - Errors may spike at kinks (e.g., borrowing constraints)
- Use visualization to:
  - Identify problematic regions requiring finer grids
  - Diagnose interpolation issues
  - Verify constraint handling

Den Haan-Marcet Statistics

- Euler errors: point-wise accuracy on evaluation grid.
- **Den Haan-Marcet (1994):** simulation-based accuracy test.
- Idea: if solution is correct, Euler equation holds on average along simulated paths.
- Construct test statistic from simulated data:

$$B_T = \frac{1}{T} \sum_{t=1}^{T} \mathcal{E}(k_t) \cdot h(k_t)$$

  where $h(\cdot)$ is a vector of instruments (e.g., $[1, k, k^2]'$).
- Under correct solution: $\mathbb{E}[B_T] = 0$.

Den Haan-Marcet: Implementation

- Simulate long path $\{k_t, c_t\}_{t=1}^{T}$ using approximate policy.
- Compute Euler residuals along path:

$$e_t = u'(c_t) - \beta u'(c_{t+1}) R(k_{t+1})$$

- Choose instruments: $h_t = [1, k_t, k_t^2, \ldots]'$
- Form statistic:

$$J = T \cdot B_T' \hat{\Omega}^{-1} B_T$$

  where $\hat{\Omega}$ is HAC estimator of variance.
- Under null (correct solution): $J \sim \chi^2(\dim(h))$.
- Reject if J too large $\Rightarrow$ solution inaccurate.

Den Haan-Marcet: Interpretation

- Advantages over Euler errors:
  - Tests accuracy where economy actually spends time (ergodic distribution)
  - Weights errors by how often states are visited
  - Formal statistical test with p-values
- Disadvantages:
  - Requires long simulation
  - May miss errors in rarely visited regions
  - Sensitive to instrument choice
- Best practice: report **both** Euler errors and DH-M statistics.

Grid Sensitivity Analysis

- Verify convergence predicted by theory (Santos 1998).
- Sensitivity analysis procedure:
  1. Solve with grids of increasing density: M, $2M$, $4M$, $8M$.
  2. Compute Euler errors on **fixed evaluation grid** for each.
  3. Check convergence rates.
- Expected rates:
  - Linear interpolation: errors $\propto h^2$ (quadratic in mesh size)
  - Cubic spline: errors $\propto h^4$
- If errors don't decrease at expected rate: check implementation, constraint handling, interpolation.

Method Comparison: Accuracy Benchmarks

- Compare methods on same problem (Aruoba, Fernández-Villaverde, Rubio-Ramírez 2006):

Method	$\log_{10}(\max \text{EE})$	Time	Grid
VFI (linear)	-3.0	10s	500
VFI (cubic)	-5.0	12s	500
Time Iteration	-5.2	8s	500
EGM	-5.1	2s	500
EGM (fine)	-7.0	5s	2000

- EGM achieves comparable accuracy in less time.
- Fine grid with EGM approaches machine precision limits.

Reporting Standards for Publications

- When publishing numerical results, report:
  1. Solution method and implementation details
  2. Grid specification (number of points, spacing, bounds)
  3. Convergence tolerance used
  4. Maximum and mean Euler equation errors **on evaluation grid**
  5. Den Haan-Marcet test results (if applicable)
  6. Sensitivity to grid density (verify theoretical rates)
  7. Computation time and hardware
- Enables replication and comparison across studies.
- Standard reference: Aruoba, Fernández-Villaverde, Rubio-Ramírez (2006).

Summary

- **Acceleration techniques:**
  - Howard's improvement: reduce iterations via policy evaluation
  - MacQueen-Porteus: tighter error bounds for early stopping
  - Monotonicity/concavity: faster per-iteration optimization
- **Endogenous Grid Method:**
  - Fix tomorrow's capital $k' \in \mathbb{K}$, solve backward for today's (c, k)
  - Eliminates root-finding; 5x–20x speedup
- **Accuracy assessment:**
  - Evaluate on separate grid (like ML train/test split)
  - Santos (1998): theoretical bounds ($O(h^2)$ for value, $O(h)$ for policy)
  - Euler errors: unit-free, interpretable
  - Den Haan-Marcet: simulation-based statistical test