Lecture VII

Value Function Iteration with Discretization

Gianluca Violante

New York University

Quantitative Macroeconomics

Stochastic growth model with leisure

- Consider the planner's problem:

$$
\begin{aligned}
V(k,z) &= \max_{c,k',h} u(c) + v(1-h) + \beta \sum_{z' \in Z} \pi(z,z') V(k',z') \\
& s.t. \\
c + k' &= e^z f(k,h) + (1-\delta) k \\
c &\geq 0 \\
k' &\geq 0 \\
h &\in [0,1]
\end{aligned}
$$

- We want to solve for the policy functions $g_c(z,k), g_k(z,k), g_h(z,k)$
- We do it by VFI + discretization of the state space (Z,K)
- V is a function defined only on grid point (a matrix)

Setting up the grids

- Grid of S points $\{z_1, ..., z_S\}$ for productivity shock (e.g., Tauchen)
- Evenly spaced grid of N points for capital:

$$k_i = k_1 + (i-1)\,\eta \quad \text{, for } i = 1, ...N \quad \text{where} \quad \eta = \frac{k_{N-}k_1}{N-1}$$

  and in general, more points where decision rules have curvature

- How do we choose $\{k_1, k_N\}$? Solve for SS system:

$$\begin{aligned} f_k\left(k^*, 1-h^*\right) &= \left(\frac{1}{\beta} - 1 + \delta\right) \\ c^* + \delta k^* &= f\left(k^*, h^*\right) \\ u_c\left(c^*\right) f_h\left(k^*, h^*\right) &= -v_h\left(1-h^*\right) \end{aligned}$$

- $\{k_1, k_N\}$ must be below and above $k^\star$. Grid size depends on the size and persistence of z. Keep k_1 away from zero.

Dealing with choice of leisure

- At every (k_i, k_j, z_s) corresponding to a grid point for the variables (k, k', z) we have the intratemporal FOC:

$$u_c\left(e^{z_s} f\left(k_i, h\right) + (1-\delta)\, k_i - k_j\right) \cdot e^{z_s} f_h\left(k_i, h\right) = -v_h\left(1-h\right)$$

- Note that the RHS is increasing in h and the LFS decreasing in h

- $h = 0$ ruled out by Inada condition on f and $h = 1$ ruled out by Inada condition on v. Interior solution between $(0, 1)$.

- Use bisection method to solve for h

- Call the solution $\tilde{g}_h\left(k_i, k_j, z_s\right)$

Algorithm

1. Define a three dimensional array R of size (N, N, S) with typical element (k_i, k_j, z_s) containing the return function

$$R(k_i, k_j, z_s) = u\left(e^{z_s} f(k_i) + (1-\delta) k_i - k_j\right) + v\left(1 - \tilde{g}_h(k_i, k_j, z_s)\right)$$

   Check whether the argument of u at point (i, j, s) is negative: if so, set $R(k_i, k_j, z_s)$ to a very large negative number.

2. Start with a guess of the (N, S) matrix V^0, say the null array so the first iteration is a static problem. Or

$$V^0(k_i, z_s) = \frac{u\left(z_s f(k_i, h^*) - \delta k_i\right) + v(1 - h^*)}{1-\beta}$$

   Denote the number of iterations by t

3. At any iteration t, you have the (N, S) matrix V^t with generic element $V^t(k_j, z_s)$

Algorithm

4. Compute the (N, S) matrix $\mathbf{V}^t$ that represents the conditional expected value with generic element

$$\mathbf{V}^t\left(k_j, z_s\right) = \sum_{s'=1}^{S} \pi\left(z_s, z_{s'}\right) V^t\left(k_j, z_{s'}\right)$$

5. Update value function by selecting

$$V^{t+1}\left(k_i, z_s\right) = \max_j \left\{R\left(k_i, k_j, z_s\right) + \beta \mathbf{V}^t\left(k_j, z_s\right)\right\}$$

   Careful here! With the $\max$ operator you are squeezing a 3-dim array into a 2-dim array. In Matlab use command called `squeeze`

Algorithm

6. Store the $\arg\max$

$$\begin{aligned}
g_k^{t+1}(k_i, z_s) &= \arg\max_j R(k_i, k_j, z_s) + \beta \mathbf{V}^t(k_j, z_s) \\
g_h^{t+1}(k_i, z_s) &= \tilde{g}_h\left(k_i, g_k^{t+1}(k_i, z_s), z_s\right) \\
g_c^{t+1}(k_i, z_s) &= e^{z_s} f\left(k_i, g_h^{t+1}\right) + (1-\delta) k_i - g_k^{t+1}(k_i, z_s)
\end{aligned}$$

7. Howard improvement step here (see later)

8. Check convergence. If $||V^{t+1} - V^t|| < \varepsilon$, stop and report success, otherwise go back to step [3.] and update V^t with V^{t+1}

Additional checks

1. Check that the policy function isn't constrained by the discrete state space. If g_k is equal to the highest or lowest value of capital in the grid for some i, relax the bounds of the grid for k and redo the value function iteration. Especially likely for upper bound.

2. Check that the error tolerance is small enough. If a small reduction in ε results in large changes in the value or policy functions, the tolerance is too high. Reduce ε until the solution to the value and policy functions are insensitive to further reductions.

3. Check that grid size is dense enough. If an increase in number of grid points results in a substantially different solution, the grid might be too sparse. Keep increasing grid size until the value and policy functions are insensitive to further increases.

Discretization is considerably slower compared with other methods that you will learn (curse of dimensionality), but it's the most robust method

Howard Improvement Step: idea

- Selecting maximizer in step [5.] to compute the policy functions is the most time-consuming step in the value function iteration
- Howard's improvement reduces the number of times we update the policy function relative to the number of times we update the value function
- Idea: on some iterations, we simply use the current approximation to the policy function to update the value function without updating the policy function
- Updating the value function using a the current estimate of the policy function will bring the value function closer to the true value since the policy function tends to converge faster than the value function

Howard Improvement Step: implementation

- Let $\mathcal{V}_0^{t+1}(k_i, z_s) = V^{t+1}(k_i, z_s)$ and iterate $h = 1, ..., H$ times:

$$\mathcal{V}_{h+1}^{t+1}(k_i, z_s) = R\left(k_i, g_k^{t+1}(k_i, z_s), z_s\right) + \beta \sum_{s'=1}^{S} \pi(z_s, z_{s'}) \mathcal{V}_h^{t+1}\left(g_k^{t+1}(k_i, z_s), z_{s'}\right)$$

  Note: g_k stays the same at each iteration h

- Need to choose H (VFI using existing policy function we perform after each policy function update)

- Too high H may result in a value function moving further from the true one since the policy function is not the optimal policy. A good idea is to increase H after each iteration. Or use the Howard algorithm only after a few steps of the VFI

Policy function iteration

1. At iteration t, g_k^t is given. Need an initial guess of policy.
2. Solve system of $N \times S$ equations and unknowns:

$$V^t\left(k_i, z_s\right) = R^t\left(k_i, g_k^t(k_i, z_s), z_s\right) + \beta \sum_{s'=1}^{S} \pi\left(z_s, z_{s'}\right) V^t\left(g_k^t(k_i, z_s), z_{s'}\right)$$

3. Update the policy function:

$$g_k^{t+1}\left(k_i, z_s\right) = \arg\max_j \left\{ R\left(k_i, k_j, z_s\right) + \beta \sum_{s'=1}^{S} \pi\left(z_s, z_{s'}\right) V^t\left(k_j, z_{s'}\right) \right\}$$

4. Go back to step 2. with new policy

PFI faster than VFI because iteration because iteration substituted with system of equations (but less stable)