Lecture II
Appendix

Adam Hal Spencer

The University of Nottingham

Applied Computational Economics 2020

Roadmap

1. Computing RCE via Policy Function Iteration

2. Computing RCE via Projection Methods

Motivation

- Consider the following twist on the neoclassical growth model we've been thinking about so far.
- Households here earn a labour income, receive dividends and can save through riskless bonds that are in zero net supply.
- The firms own the capital stock and invest in it optimally each period.
- Let's see what we can say about a competitive equilibrium in this context...

Motivation

- Household's problem looks like the following

$$\max_{\{c_t, b_{t+1}, n_t\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t, n_t)$$

subject to

$$c_t + b_{t+1} \leqslant w_t n_t + b_t(1+r) + d_t$$

Motivation

- Firm's problem looks like the following

$$\max_{\{k_{t+1},n_t,d_t\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_{t=0}^{\infty} M_t d_t$$

where

$$d_t = a_t k_t^{\alpha} n_t^{\gamma} - k_{t+1} + (1-\delta)k_t - w_t n_t$$

and a_t is some stochastic process.

- What the hell is M_t?

Motivation

- Firm pays dividends to the households.
- Firms' investment decision-making should account for the preferences of the households.
- We should discount using the household's preferences:

$$M_t = \beta^t u'(c_t)$$

- Relevant due to incomplete markets setup: households value consumption differently in different states of the world.

Motivation

- We can write the Bellman equation for the firm as

$$V(a,k,M) = \max_{k',n,d} \; M[ak^{\alpha}n^{\gamma} - k' + (1-\delta)k - w_t n_t] + \mathbb{E}[M'V(a',k',M')]$$

Motivation

- What's the complicating factor here?
- The firm's decisions must all be with the household's SDF as given!
- It (or consumption c_t) becomes a state in the firm's recursive formulation.
- This is not straightforward to solve using value function iteration.
- Firm takes c_t as given, (prehaps) with a law of motion similar to the "big K-little k" setup, optimise, check consistency....it's a mess.

Motivation

- Given that the household's utility function is concave, the firm's FOCs are both necessary and sufficient for a solution to the problem.
- We can work with this.
- Leverage this to instead iterate on the policy function using the Euler equation.

Motivation

- The firm's Euler equation (in sequence form) is

$$M_t = \mathbb{E}_t \left[M_{t+1} (\alpha a_{t+1} k_{t+1}^{\alpha-1} n_{t+1}^{\gamma} + \{1 - \delta\}) \right]$$

where there's also an intra-temporal choice of labour input.

- This Euler equation is then given by

$$u'(c_t) = \beta \mathbb{E}_t \left[u'(c_{t+1}) (\alpha a_{t+1} k_{t+1}^{\alpha-1} n_{t+1}^{\gamma} + \{1 - \delta\}) \right].$$

Motivation

- This is the same Euler equation as if we solved the problem with the households instead owning the capital stock.
- So we could re-write the problem and solve it indirectly via value function iteration on the household's problem.
- We'll assume though that you want to solve the problem directly, (i.e. as is: with the firm owning the capital stock).
- For more complicated problems, you may have no choice but to solve it directly.

Functional Equations

- We can usually re-write the optimality conditions in one of our problems in the form

$$\mathbb{E}_{a'}[f(x, x', x'', a, a')] = 0$$

and we're usually looking for a policy function of the form $x' = g(x, a)$ such that

$$\mathbb{E}_{a'}[f(x, g(x, a), g(g(x, a), a'), a, a')] = 0.$$

- We seek an approximation to the policy function $x' = g(x, a)\ \forall x \in \mathscr{X}, a \in \mathscr{A}$.

Solving Functional Equations (1): Time Iteration

- Start with some guess: a candidate policy function $g_n(x, a)$.
- Plug it in to the functional equation and solve for x' such that the equality holds

$$\mathbb{E}_{a'}[f(x, x', g_n(x', a'), a, a')] = 0$$

where notice that here we're plugging $g_n(x', a')$ and solving for the new policy function $g_{n+1}(x, a)$.

- Can use a nonlinear equation solver.
- Keep iterating until

$$||\mathbb{E}_{a'}[f(x, g_n(x, a), g_n(g_n(x, a), a'), a, a')]|| < \epsilon$$

Solving Functional Equations (1): Time Iteration

- If the original problem was a contraction mapping, (e.g. Bellman equation), then this procedure is the same as VFI.
- Convergence properties the same.

Solving Functional Equations (2): Fixed Point Iteration

- Sometimes it's possible to re-write the functional equation in the form

$$\mathbb{E}_{a'}[f(x, x', x'', a, a')] - x' = 0$$

for all $x \in \mathscr{X}$ and $a \in \mathscr{A}$.

- If we have a candidate $g_n(x, a)$ then we can update using

$$x' = \mathbb{E}_{a'}[f(x, g_n(x, a), g_n(g_n(x, a), a'), a, a')]$$

- Continue iterating until

$$||x' - \mathbb{E}_{a'}[f(x, g_n(x, a), g_n(g_n(x, a), a'), a, a')]|| < \epsilon$$

Solving Functional Equations (2): Fixed Point Iteration

- This algorithm can be fast, really fast.
- Convergence might be problematic though.
- Sometimes good to update the policy function "slowly".
- Meaning, set $g_{n+1}(x, a) = \omega g_n(x, a) + (1 - \omega)x'$ for some $\omega \in [0, 1]$.

Solving Functional Equations (3): Endogenous Grids

- So far our approach has always started with discretising our state space.

- Then for each value in the resulting grid, we'd find a corresponding value of the policy function.

- E.g. recall the Euler equation and resource constraint for the stochastic growth model

$$u'(c) = \beta \mathbb{E}[u'(c')[1 - \delta + \alpha(a')(k')^{\alpha-1}]]$$
$$c = ak^{\alpha} - k' + (1 - \delta)k$$

- Then we'd discretise the set for k into $\{k_1, k_2, ..., k_N\}$ and a into $\{a_1, a_2, ..., a_N\}$ in the stochastic growth model and then find $k'(k, a)$ corresponding to each gridpoint pair.

Solving Functional Equations (3): Endogenous Grids

- The method of endogenous gridpoints flips the problem around.
- Create a grid for x' instead of x.
- Solve for x from

$$\mathbb{E}_{a'}[f(x, x', g_n(x', a'), a, a')] = 0.$$

then use the discretised vector for x' with the solution for x to update the policy function $g_{n+1}(x, a)$.

Solving Functional Equations (3): Endogenous Grids

- E.g. back to the stochastic growth model.
- Create a grid for k' as $\{k'_1, k'_2, ..., k'_N\}$, (still do the same for a).
- Define new state variable y as

$$y = ak^{\alpha} + (1 - \delta)k$$

where

$$u'(y - k') = \beta\mathbb{E}[u'(y' - g_n(y', a'))\{1 - \delta + \alpha(a')(k')^{\alpha-1}\}]$$

Solving Functional Equations (3): Endogenous Grids

- Then assuming invertability of the utility function, we can say

$$y = u'^{-1}\left(\beta\mathbb{E}[u'(y' - g_n(y', a'))\{1 - \delta + \alpha(a')(k')^{\alpha-1}\}]\right) + k'$$

  gives us the k such that the choice of k' is optimal for that pair (k, a).

- Rather than finding the k' that corresponds with the k grid, the procedure is telling us the k that corresponds with the choice of k'.

- Update $g_{n+1}(y, a)$ accordingly.

Roadmap

1 Computing RCE via Policy Function Iteration

2 Computing RCE via Projection Methods

Motivation

- We've already touched on this a bit in the last lecture.
- We can use interpolation methods to approximate a continuous value function using monomials or, better yet, orthogonal polynomials.
- Another alternative is to approximate policy functions in this way.
- I won't spend much time on it, but I'll give a quick example to make it clear.
- Let's just keep it simple: we'll use monomials.

Projections of Policy Functions

- Let's think about the deterministic neoclassical growth model.
- Say there's inelastic labour supply and CRRA preferences.
- The solution is given by the Euler equation and resource constraint respectively:

$$c^{-\sigma} = \beta(c')^{-\sigma}\{\alpha(k')^{\alpha-1} + (1-\delta)\}$$
$$k' = k^{\alpha} - c + (1-\delta)k$$

- Since this is deterministic, our only state variable is k.

Projections of Policy Functions

- Let's consider approximation with a second order polynomial.
- Denote the vector of coefficients as $\vec{b} = (b_0, b_1, b_2)$.
- Let our policy function take the form of

$$c = \mathscr{c}(k, \vec{b}) = \exp(b_0 + b_1 \log(k) + b_2 \log(k^2))$$

why the exponential-log trickery?

Projections of Policy Functions

- We'll take as our criterion function to be the sum of squared residuals

$$\min_{\{\vec{b}\}} \sum_{i=1}^{n} R_i^2$$

  where i denotes an index over a grid of states, (to be defined shortly).

- We can use many other criteria instead: this feels natural; like an OLS analogue.

Projections of Policy Functions

- Procedure is then

(1) Choose a grid of the state $\{k_1, k_2, ..., k_n\}$ and an initial guess for the coefficient vector.

(2) For each gridpoint, compute the following objects

$$k_i' = k_i^\alpha - c(k_i, \vec{b}) + (1-\delta)k_i$$
$$c(k_i, \vec{b}) = \exp\left(b_0 + b_1 \log(k_i) + b_2 \log(k_i^2)\right)$$
$$R_i = c(k_i, \vec{b})^{-\sigma} - \beta c(k_i', \vec{b})[\alpha(k_i')^{\alpha-1} + 1 - \delta]$$

(3) Choose the new coefficients to mimimise

$$\sum_{i=1}^{n} R_i^2$$

Projections of Policy Functions

- Where is this form of the residual function coming from?
- Non-linear least squares!
- I often have trouble getting these projection approaches to policy function approximation to converge.