Aiyagari Model: Endogenous Grid Method

Yanran Guo

October 17, 2019

The goal of this note is to illustrate how to solve a standard incomplete market models (a la Aiyagari-Hugget) using the Endogenous Grid Method (EGM).

1 Model Setup

$$
\begin{aligned}
V(a,s) &= \max_{c,a'} \frac{c^{1-\gamma}-1}{1-\gamma} + \beta \mathbb{E}\big[V(a',s')|s\big] \\
s.t.\ \ & c + a' = ws + (1+r)a, \quad a' \geq \phi \\
& \log s_t = \rho \log s_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,\sigma^2)
\end{aligned}
$$

FOC w.r.t. a'

$$c^{-\gamma} = \beta \mathbb{E}\big[V_a(')\big] + \mu$$

Envelope condition

$$V_a = (1+r)c^{-\gamma}$$

E.E.

$$c^{-\gamma} = \beta(1+r)\mathbb{E}\big[c'^{-\gamma}\big] + \mu$$

2 Algorithm Summary

In EGM, we guess $c'(a',s')$ on the future-asset grid a' for each s'. Then

- Use Euler to back out current $c(a',s)$ for each pair (a',s)

$$c(a',s) = \big[\beta(1+r)\sum_{s'}\pi(s,s')u'(c'(a',s'))\big]^{-1/\gamma}$$

- Convert $(c(a',s),a')$ into the endogenous current asset.

$$a = \frac{c(a',s) + a' - ws}{1+r} \equiv a_{endo}(a',s)$$

- For each s, invert (by sorting & interpolating) the mapping $a \to a'(a, s)$ to get the saving policy $a'(a, s)$
  Note that we can either interpolate to get the saving policy or directly interpolate to get the consumption policy. Both ways work.

- Recover $c(a, s) = ws + (1 + r)a - a'(a, s)$
  Also handle the borrowing constraint region where the mapping doesn't reach the exogenous $a_{min} = \phi$

3 Numerical Implementation

1. Guess policy function for consumption: cPolM_temp is $na \times ns$ matrix, which is the combination of each a and each s.

```
cPolM_temp = repmat(aGridV, 1, ns) * (1+r) + repmat(sGridV', na, 1) * w;
```

   Note that the "repmat(aGridV, 1, ns)" in generating cPolM_temp is "each current asset holding".

2. We use this guessed policy function for c'. That means for each possible combination of a' and s', we know the corresponding consumption c'.

3. Given c', we can compute the correpsonding current consumption c using the E.E.
   Note that the c we are computing here is "when the future asset is a' and the current productivity is s, what's the correpsonding current consumption"

```
uc_next = max(cPolM_temp, 1e-12).^(-gamma);
exp_uc_next = beta * (1+r) * uc_next * sProbM';
cEndoM = max(exp_uc_next .^ (-1/gamma), 1e-12);
```

   Note that when I compute uc_next, instead of using cPolM_temp directly, I use max(cPolM_temp, 1e-12). This avoids the situation where some elements in cPolM_temp is negative or zero, leading to -Inf for uc_next. The same logic applies to the compuation of cEndoM.

4. cEndoM is the combination of each a' and s. And the future asset holding (which is the saving choice in the current period) is just aGridV. Therefore, using cEndoM(a',s) and the guessed policy function for saving, I can calculate the corresponding current asset holding using the budget constraint.

```
aEndoM = (cEndoM + aGridV - w * sGridV') / (1 + r);
```

   Note that in EGM, we are using aGridV for a'.

5. Now we can do interpolation. The saving decision, which in this algorithm is just aprimeM, is based on each column in aEndoM and a given s. That's the policy function for saving.

```
aPolM = zeros(na, ns);
for is = 1 : ns
aPolM(:, is) = interp1(aEndoM(:, is), aGridV, aGridV);
end
```

The interpolation is conducted under the assumption that we have a non-binding borrowing constraint. Now we need to take care of the regions where the borrowing contraint is binding.

```
aEndoMinV = min(aEndoM, [], 1);
bindMask = aV < aEndoMinV;
aPolM(bindMask) = phi;
aPolM = max(aPolM, phi); % hard LB everywhere (safety)
```

6. Now we can recover $c(a, s)$ from budget constraint, because we have the policy function for saving, aPolM, and the total income given (a, s).

```
cPolM = max(w*sGridV' + (1+r)*aGridV - aPolM, 1e-12)
```

7. Now we generate the distance between cPolM and cPolM_temp to check convergence. If the difference between the two matrices is less than the tolerance level, then stop. Otherwise, update the guess for policy functions for consumption and saving, and then go back to Step 3.

```
diff = max(abs(cPolM(:) - cPolM_temp(:)));
cPolM_temp = cPolM;
```