Robert Ulbricht
Boston College

Problem set 1

Due Wednesday, Sep 7, 11.59pm by email. In addition to your codes, please compile a pdf that contains all requested figures and numbers.

---

1 Steady state of the standard incomplete markets model

Consider the standard incomplete markets model from the lecture with the following parametrization:

- $u(c) = \log c$
- $\log s_{it} = \rho \log s_{it-1} + \epsilon_{it}$ with
  - $\epsilon_{it} \sim \mathcal{N}\left(0, \sigma_\epsilon \sqrt{1-\rho^2}\right)$
  - $\rho = 0.975$ and $\sigma_\epsilon = 0.7$
- $Y = 1$, $r = 0.01/4$, $\underline{a} = 0$, $\beta = 0.988$

Write a program that solves the Bellman equation and the steady state distribution! Specifically, do the following:

1. Set up grids for a and s, and define the transition matrix $\Pi_{ss'}$. Specifically:

   (a) discretize s using Rouwenhorst's method to 7 grid points, yielding the grid for s and $\Pi_{ss'}$. Normalize the resulting grid so that $\mathbb{E}_i s_{it} = 1$.

   (b) discretize a to 500 grid points between $\underline{a} = 0$ and $a^{\max} = 200$, using the geometric spacing + pivoting from the lecture.

   If you use Python, you can download the library `utils.py` from the `codes/` folder in my Dropbox repository and use the following code to set up the grids:

```
import utils  # import utils.py (should be in same directory as this code)

num_s, num_a = 7, 500  # set num_s = 7 and num_a = 500
amax = 200  # largest grid point for a
rho, sigma = 0.975, 0.7  # persistence and unconditional s.d. for log(s)

# use functions defined in utils.py to set up the grids
s_grid, pi_s, Pi = utils.markov_rouwenhorst(rho, sigma, num_s)
a_grid = utils.agrid(amax, num_a, amin=0)
```

2. Write a function `backward_step` that takes $\partial V/\partial a$ as input, and returns $\partial V/\partial a_-$ as well as the savings and consumption policies $a(s, a_-)$ and $c(s, a_-)$. Use the endogenous gridpoint method discussed in class to do so. In Python, you can use the provided function `utils.interpolate_y` to do the inverting/interpolation from $a_-(s, a)$ to $a(s, a_-)$. (In Matlab, there are standard functions to do so).

   Hint: besides $\partial V/\partial a$, you will also need to pass various parameters to `backward_step`, including the transition matrix $\Pi_{ss'}$, the grids for a and s, the interest rate r, aggregate income Y, discount factor β, and—optionally—the intertemporal elasticity of substitution σ in case you choose to implement a more general version where $u(c) = \frac{1}{1-\sigma^{-1}} c^{1-\sigma^{-1}}$.

3. Initialize $\partial V/\partial a_-$ guessing that for each (s, a_-) households consume 10% of their available "cash on hand":

$$c(s, a_-) = 0.1 \cdot ((1 + r)a_- + sY).$$

   Then substitute this guess into the envelope condition to obtain an initial guess for the marginal value of assets:

$$\frac{\partial V}{\partial a_-}(s, a_-) = (1 + r)u'(c(s, a_-)).$$

4. Starting from this initial guess for $\partial V/\partial a_-$, iterate on `backward_step` until convergence.

5. Plot the resulting net saving policy, $a(s, a_-) - a_-$, and consumption function $c(s, a_-)$ as a function of a_-. (Your plots should have 7 lines, one for each income level).

Congratulations, you have solved the Bellman equation! Next, use the resulting savings policy $a(s, a_-)$ to compute the stationary distribution over (s, a).

6. Use Young's method to discretize the savings policy $a(s, a_-)$ that solves the Bellman equation. Specifically, for each point in the state space (s, a_-), the discretization should give you an index $\mathtt{i_a}$ and weight π_a on the lower gridpoint bracketing $a(s, a_-)$, such that:

$$\mathtt{a} = \pi_\mathtt{a} \cdot \mathtt{a\_grid}[\mathtt{i_a}] + (1 - \pi_\mathtt{a}) \cdot \mathtt{a\_grid}[\mathtt{i_a} + 1].$$

   If you use Python, you can use the provided function `utils.youngs_method` to do so, using the following code:

```
a_i, a_pi = utils.youngs_method(a_grid, a)  # a is optimal savings policy
```

7. Write a function `forward_step` that takes a probability mass distribution over (s, a_-) as input, and returns a probability mass distribution over (s', a).

   Hint: In addition to the current guess for the distribution D_t, you will also need to pass the transition matrix $\Pi_{ss'}$ and the discretized asset policy (i_a, π_a) from the previous step as parameters to `forward_step`. To arrive at D_{t+1}, it's best to proceed in 2 steps:

   (a) Initialize $D_t^{\text{end}} = \mathbf{0}_{\mathtt{num\_s} \times \mathtt{num\_a}}$. Then update the asset position by iterating through all (s, a_-). Letting $(\mathtt{i_s}, \mathtt{i_{a_-}})$ denote the corresponding indices of the grid, for each $(\mathtt{i_s}, \mathtt{i_{a_-}})$:

      i. fix $D = D_t(\mathtt{i_s}, \mathtt{i_{a_-}})$ as the mass of the originating state.

      ii. look up the index $\mathtt{i_a}$ and weight π_a for next periods' asset position at $(\mathtt{i_s}, \mathtt{i_{a_-}})$.

      iii. then add $\pi_a \cdot D$ to $D^{\text{end}}(\mathtt{i_s}, \mathtt{i_a})$ and add $(1 - \pi_a) \cdot D$ to $D^{\text{end}}(\mathtt{i_s}, \mathtt{i_a} + 1)$.

(b) Update the income to arrive at D_{t+1} by premultiplying D_t^{end} from the previous step with the transpose of $\Pi_{ss'}$. In Python:

```
D_next = Pi.T @ D_end
```

8. Initialize some initial distribution over (s, a) and iterate on forward_step until convergence. A decent initial guess for D is to assume that a and s are independent, with a being uniformly distributed across its grid, and s being distributed according to its stationary distribution. (If you used my code above to implement Rouwenhorst's method, the stationary distribution over s is given by pi_s).

9. Use the resulting invariant distribution to compute:

(c) the aggregate asset stock, normalized by annual earnings,

$$A = \sum_{s,a_-} D(s, a_-) \cdot \frac{a_-}{4Y},$$

(d) the average marginal propensity to consume out of incoming assets a_-

$$\overline{MPC} = \sum_{s,a_-} D(s, a_-) \cdot \frac{\partial c(s, a_-)}{\partial a_-}.$$

(*Hint:* Use forward differences on a_grid to numerical differentiate the consumption policy.)

10. How do the aggregate asset stock and the average MPC vary with β?

11. Plot a histogram of the cross-sectional asset distribution.

Congratulations, you solved the standard incomplete markets model. Feel free to play around with your code and explore other parametrization and implications. Bonus question: How long does it take to solve for the steady state on your computer? In Python, you can use the provided tic-toc functions to time your code:

```
utils.tic()
# add your code here
utils.toc()
```

2 Homogeneity of the aggregate savings functions

Prove that the aggregate asset stock A is homogenous of degree 1 in $(Y, \underline{a})$ for $u(c) = \frac{c^{1-\gamma}-1}{1-\gamma}$ with $\gamma > 0$.

Hint: This question is intended to be done by pen & paper. If you are trying to verify this on your computer, you will also need to adjust the upper bound of the asset grid accordingly.