Stochastic Multi-Period OLG Model Computation

Prof. Lutz Hendricks

Econ821

February 5, 2015

Contents

Program Outline

Set constants:

- exogenous parameters
- guesses for calibrated parameters: (A, δ, β).
- `const_ogm`

Set parameters that do not require solution of household problem:

- $A, \delta \rightarrow w$ and r targets (given K/Y).
- Capital grid.
- Markov chain for labor endowments $\rightarrow$ approximate AR(1).
- `param_set_ogm`

Precompute labor endowment histories.
Precompute aggregate labor supply (exogenous).
Find β that matches K/Y target: `cal_dev_ogm`.

Calibration algorithm

`cal_dev_ogm`

For each β guess:

1. Solve **household** problem $\rightarrow$ policy functions `cPolM(ik,ie,a)` and `kPolM(ik,ie,a)`.
2. **Simulate** a large number of households (k histories: `kHistM(a, ind)`).
3. Compute **aggregate** K and Y from simulated histories.
4. Return deviation from target K/Y.

Household Problem

Solve for policy functions by **backward induction**: `hh_solve_ogm`
In last period (age a_D) household consumes all income: $c(k, s) = y(k, s)$.
At earlier ages (a): `hh_solve_age_ogm`

- Take policy function for $a + 1$ as given.
- For each state (k, s):
  - Search over values of c that zero the Euler equation deviation (`hh_opt_c_ogm`).
  - Store the optimal choice in a matrix `cPolM(ik,ie,a)`.

Finding zero of Euler equation for one state: `hh_opt_c_ogm`.

- Search over Euler equation deviations (`hh_ee_dev_ogm`).
- Use precomputed expected marginal utility when old.
- Complication: Must first check that household does not choose a corner ($k' = 0$).

Euler equation deviation

for one state and k': `hh_ee_dev_ogm`

- Compute c from the budget constraint: $c = y - k'$.
- For each possible state tomorrow (e') compute $u'(c'[e'])$.
  - Take c' from tomorrow's policy function `cPolM`. This requires interpolation because k' is not on the grid.
- Compute expected marginal utility tomorrow:

$$E\{u'(c')\} = \sum_{e'} \Pr(e'|e)\ u'(c'[e'])$$

- Return deviation: $u'(c) - \beta\ R'\ E\{u'(c')\}$. Transform to avoid non-linearity.

This is very slow.

Approximation errors are big, unless k grid is very find at low k

How to make it faster?

Household: Value Function Iteration

A more accurate solution.

`hh_solve_vfi_ogm`

Finding optimal k'

IN:

- y, R, e, parameters
- continuous approximation of $\mathbb{E}V\left(k'; e', a+1\right)$

OUT: $k', c, V\left(k, e, a\right)$

Steps:

1. Set feasible range for k'
2. If no k' feasible, set $k' = kGrid(1)$
3. Set up Bellman operator
4. Use `fminbnd` to find max of Bellman

Bellman Operator

`hh_optc_vfi_ogm`

1. $c = \max\{cFloor, y - k'\}$
2. $V = u(c) + \beta R \mathbb{E} V(k'; e', a+1)$

Algorithm Details

Stationarity

There is no need to ensure that the household distribution is **stationary**.

- The reason is that all household endowments are exogenous (k_1, e_1).
- If each generation faces the same prices, they will make the same choices.
- This changes when households receive inheritances or human capital investments from their parents.
- Then: Iterate over household simulations until distribution becomes stationary.

Capital Grid

Number of grid points: Must be set such that quality of approximation is sufficiently good.
But increasing n_k is computationally costly.
We set $n_k = 50$ for starters.

Top capital value:

- Must be set such that no household ever reaches it.
- Start with a guess.
- Later check that it is not (rarely) binding.

It would be more efficient to have a different grid for each age (young households cannot hold as much wealth as old ones).

Simulating household histories

Need to draw **random numbers** (realizations of earnings shocks).

- `randn` draws Gaussian random numbers.
- It is important to use the same random numbers for every iteration over β guesses.
- Otherwise simulated aggregates change a little bit every time which confuses equation solvers.

Simulating Markov chains:

- Programs for doing this are in `shared` directory.
- `markov_cohort_sim` takes a transition matrix $\Pr(e'|e)$ and a vector of age 1 states, then simulates e histories for a large number of households.

Computing aggregates

`aggr_hist_ogm`.

Given a history of, say, individual capital holdings, `kHistM(ind, age)`, compute the aggregate capital stock.

Because the economy is stationary, we can treat the entire history as one cross-section.

That is: we think of `kHistM(:, a)` as the cohort aged `a` today.

Let the mass of age a households be $\mu(a)$. In our model: $\mu(a) = 1/a_D$. Then

$$K = \sum_{a=1}^{a_D} \mu(a) \;\; mean\left(kHistM(:, a)\right)$$

Writing the Code

Start with primitives:

- $u'(c)$ and its inverse: `ces_util_821`
- production function: `prod_fct_ogm`

Computational primitives:

- capital grid: `kgrid_ogm`
- aggregation from histories: `aggr_hist_ogm`
- calibrating the labor endowment process: `cal_earn_ogm`
- household income: `hh_income_ogm`

Household Code

Start from inside out.

EE deviation: Easy

Optimal c, given $\mathbb{E}u'(c')$ for each k': Tricky - need to consider corner solutions.
Write out pseudo-code...

Steady State Properties

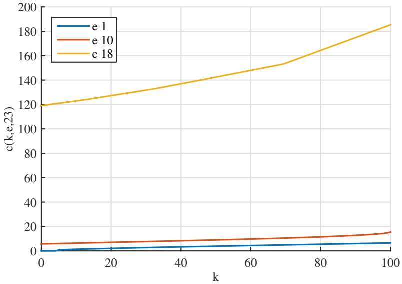
The programs save:

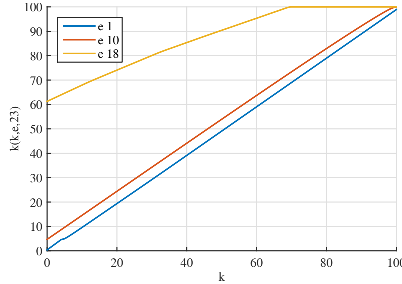
- Simulated histories for `nSim` households: `cHistM(ind, age)`, `kHistM(ind, age)`, `lsHistM(ind, age)`
- Aggregates: K, Y, L, etc.

To generate summary statistics: treat the simulated households like an actual dataset.

- `bg_stats_ogm`

Policy Functions





Comparison with Huggett (1996)

Cross-sectional wealth distribution

Gini: 0.50

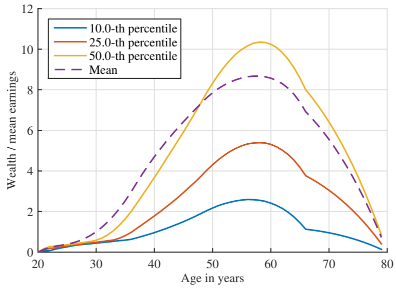
Fraction held by top 1 pct: 2.6 pct

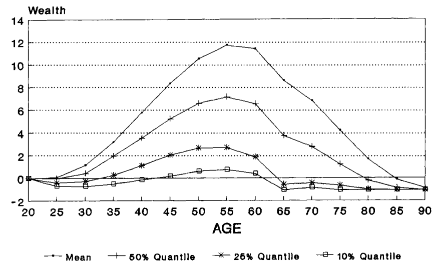
Fraction held by top 5 pct: 12.9 pct

Fraction held by top 25 pct: 58.1 pct

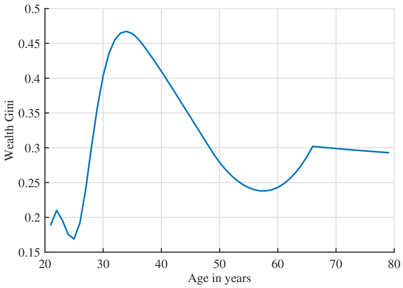
Fraction held by top 50 pct: 87.9 pct

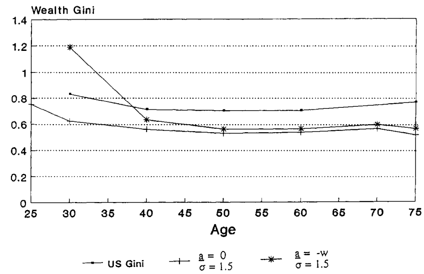
Age wealth profiles





Wealth Ginis by age





Exercise

What other statistics would one like to match?

- Write some code to compute those statistics.

Check that the earnings process approximates the target AR(1)

- To estimate an AR(1), match the auto-covariance matrix (Guvenen)

Extensions

Ex ante heterogeneity

Example: households differ in risk aversion or discount factors

Assume there are J types: $j = 1, ..., J$ with mass m_j.

$\sum_j m_j = N$

Assignment: Modify the code for this case.

We will talk in the next class about any difficulties you encounter.

Note: Be generic.

- Even if households differ in several endowments, just call each combination a type j.
- Then your code does not depend on the nature of heterogeneity.

Intergenerational Links

A simple case: stochastic mortality.

Assume that assets of dying households are given to living households as lump-sum transfers (e.g. everyone gets the same amount)

What changes:

- Household discounts at $\beta\times$survival probability
- Mass of households by age changes
- That affects code for computing aggregates
- Now we need to iterate over a guess for the lump-sum transfer in addition to β

Bequests

Households leave their terminal wealth to newly born agents (generations do not overlap). What changes:

- Now we need to iterate over a guess for the distribution of inheritances (in addition to β)

References