Algorithm

Nicolas Fernandez-Arias

March 17, 2018

1 Outline

At the highest level, the algorithm for computing the equilibrium of the economy consists of the following iterative scheme.

1. Guess final goods labor g
   (a) Guess growth rate L^F
      i. Guess R&D wages $w(q, m)$
         A. Guess entrant innovation effort $z^E(q, m)$
         B. Compute V using HACT method by Moll et al.
         C. Compute W using same method
         D. Aggregate individual policies from computation of W to compute implied $\tilde{z}^E(q, m)$.
         E. If not converged, update guess and return to (1aiA)
      ii. Compute wage $\tilde{w}(q, m) = \overline{w} - \nu W(q, m)$
         - If not converged, update guess and return to (1ai)
   (b) Compute stationary distribution, aggregate policy functions to implied $\tilde{L}^F$
      - If not converged, update guess and return to (1a)
2. Using stationary distribution and policy functions, compute implied $\tilde{g}$
   - If not converged, update guess and return to (1)

2 Finite difference solution of HJBs

2.1 Incumbent

2.1.1 Explicit method

Define Δ_t

2.1.2 Semi-implicit method

In order to make this work, need to be smart about what grid points we use for q. Essentially, need it to be possible to compute V^+ without explicit reference to q, only to the index i_q. This can be achieved by having the points on the grid log-spaced; specifically, $q_{i+1} = q_i * (1+\lambda)^{1/m}$ for some $m \geq 1$. Set $m > 1$ to make the grid finer.

Discretization:

$$\frac{V_{i,j}^{n+1} - V_{i,j}^n}{\Delta_t} + (\rho - g)V_{i,j}^{n+1} = f_{i,j}^n + \xi_{i,j}^n V_{i,j}^{n+1} + \eta_{i,j} V_{i-1,j}^{n+1} + \gamma_{i,j} V_{i,j+1}^{n+1} + \sigma_{i,j}\Big(V^{n+1}\Big)_{i,j}^+$$

with

$$\begin{aligned}
f_{i,j}^n &= \pi_{i,j} + x_{i,j}^n(-\overline{w} z_{i,j}^n) + (1 - x_{i,j}^n)(-w_{i,j} z_{i,j}^n) \\
\xi_{i,j}^n &= -gq/\Delta_i^q - (\chi_I z_{i,j}^n + \chi_E z_{i,j}^E)\phi(z_{i,j}^n + z_{i,j}^E)h(q_i) \\
\eta_{i,j}^n &= gq/\Delta_i^q \\
\gamma_{i,j}^n &= \nu(z_{i,j}^E + (1 - x_{i,j}^n)z_{i,j}^n)/\Delta_i^m \\
\sigma_{i,j}^n &= \chi_I z_{i,j}^n \phi(z_{i,j}^n + z_{i,j}^E)h(q_i) \\
\Delta_i^q &= q_i - q_{i-1} = (1 - (1+\lambda)^{-1/m})q_i \\
\Delta_i^m &= m_{i+1} - m_i
\end{aligned}$$

Further, we compute $\Big(V^{n+1}\Big)_{i,j}^+$, extrapolating linearly when necessary:

$$\Big(V^{n+1}\Big)_{i,j}^+ = \begin{cases} V_{i+m,0}^{n+1} & i < I - (m-1) \\ V_{i,0}^{n+1} + \frac{\lambda}{(1+\lambda)^{1/m} - 1}\Big(V_{i+1,0}^{n+1} - V_{i,0}^{n+1}\Big) & I - (m-1) <= i < I \\ V_{i,0}^{n+1} + \frac{\lambda}{1 - (1+\lambda)^{-1/m}}\Big(V_{i,0}^{n+1} - V_{i-1,0}^{n+1}\Big) & i = I \end{cases}$$

Grouping like terms, have the expression

$$
\begin{aligned}
(V^{n+1})^{+}_{i,j} &= \mathbb{1}_{\{i<I-(m-1)\}} V^{n+1}_{i+m,0} \\
&+ \mathbb{1}_{\{I-(m-1)\leq i<I\}}\Big(\Big(1-\frac{\lambda}{(1+\lambda)^{1/m}-1}\Big)V^{n+1}_{i,0} + \frac{\lambda}{(1+\lambda)^{1/m}-1}V^{n+1}_{i+1,0}\Big) \\
&+ \mathbb{1}_{\{i=I\}}\Big(\Big(1+\frac{\lambda}{1-(1+\lambda)^{-1/m}}\Big)V^{n+1}_{i,0} - \frac{\lambda}{1-(1+\lambda)^{-1/m}}V^{n+1}_{i-1,0}\Big)
\end{aligned}
$$

3 Computing the stationary distribution

3.1 Simulation

3.2 Kolmogorov Forward Equations

The Kolmogorov Forward equation is (need to prove)

$$
\begin{aligned}
0 &= -\partial_q\Big(a^q(q,m)\mu(q,m)\Big) - \partial_m\Big(a^m(q,m)\mu(q,m)\Big) \\
&- s(q,m)\mu(q,m) \\
&+ \mathbb{1}_{\{m=0\}}\int_0^\infty s((1+\lambda)^{-1}q,m')\mu(q,m')dm'
\end{aligned}
$$

where $a^q(q,m), a^m(q,m)$ is the drift in the q,m direction, respectively, and $s(q,m)$ is the aggregate innovation rate for a product in state (q,m). Have

$$
\begin{aligned}
a^q(q,m) &= -gq \\
a^m(q,m) &= \nu(z^E(q,m) + (1-x(q,m)z^I(q,m))) \\
s(q,m) &= (\chi_I z^I(q,m) + \chi_E z^E(q,m))\phi(\tau(q,m)) \\
\zeta(q,m) &= z^E(q,m) + (1-x(q,m))z^I(q,m)
\end{aligned}
$$

Following Moll's HACT numerical appendix, discretize the HJB as

$$
\begin{aligned}
0 = &-\frac{-gq_{i+1}\mu_{i+1,j} + gq_i\mu_{i,j}}{\Delta^q_{i+1}} - \frac{\nu\zeta_{i,j-1}\mu_{i,j-1} - \nu\zeta_{i,j}\mu_{i,j}}{\Delta^m_{i-1}} \\
&- s_{i,j}\mu_{i,j} + \mathbb{1}_{\{j=1\}}\sum_{j'=1}^{J} s_{i-m,j'}\mu_{i-m,j'}
\end{aligned}
$$

Finally, collecting terms, and using $\zeta_{i,j} = z^E_{i,j} + (1 - x_{i,j})z^I_{i,j}$, have

$$\begin{aligned} 0 = &-\nu\zeta_{i,j-1}\mu_{i,j-1} + gq_{i+1}\mu_{i+1,j} \\ &+ (-gq_i + \nu\zeta_{i,j} - s_{i,j})\mu_{i,j} \\ &+ \mathbb{1}_{\{j=1\}} \sum_{j'=1}^{J} s_{i-m,j'}\mu_{i-m,j'} \end{aligned}$$

Following Moll, when the subscripts are not in the state space, $\mu = 0$ so we just ignore those terms.

To put these equations in matrix form, we cannot use the matrix A as in Moll because $(V^{n+1})^+_{i,j}$ is computed by extrapolation. But I can use the matrix C^n defined by removing the contribution from the extrapolation. We can compute C in the same way as A but setting

$$\left(V^{n+1}\right)^+_{i,j} = \begin{cases} V^{n+1}_{i+m,0} & i < I - (m-1) \\ 0 & i >= I - (m-1) \end{cases}$$

As argued in Moll, if we use the matrix C from above, the stationary distribution is given by the solution to the eigenvalue problem

$$C^T\mu = 0$$

In Moll, the eigenvalue is normalized so that it sums to one. We are ignoring some points that are actually in the state space. We can, in theory, compute this mass as well, simply assuming that

4 Update rules for $g, L^F, w(q,m), z^E(q,m)$

4.1 Rule for $z^E(q,m)$

Naive approach is to simply set

$$\tilde{z}^E(q,m) = \mathbb{1}_{z^e(q,m)>0}\xi m$$

and then update according to

$$z^E_1(q,m) = \alpha z^E_0(q,m) + (1-\alpha)\tilde{z}^E_0(q,m)$$

But this has problems converging, because the policy jumps around too much, leading to

instability. Ideally one would want to vary the rate of updating in some way to compensate for this. I have tried one way, which works with $h(q) = 1$ but has had some issues for non-trivial $h(q)$.