Advanced Tools in Macroeconomics

Continuous time models (and methods)

Pontus Rendahl

August 25, 2017

Plan for today

1. Speeding things up while making them more robust
   1.1 Use the "implicit method" to somewhat bring back the contraction property
   1.2 Make use of sparsity
2. Set up a continuous time heterogenous agents model and show how to solve it
3. Kolmogorov Forward equation
4. Equilibrium

Why is the contraction property lost?

- Consider the deterministic Ramsey growth model again

$$v(k) = \max_c \{u(c) + (1-\rho)v(f(k) + (1-\delta)k - c)\}.$$

- In discrete time we iterate as

$$v_{n+1}(k) = \max_c \{u(c) + (1-\rho)v_n(f(k) + (1-\delta)k - c)\},$$

- This is a contraction mapping and we know that $v_n \to v$.

Why is the contraction property lost?

- Let's, heuristically, convert this into continuous time

$$v_{n+1}(k) = \max_c \{\Delta u(c) + (1 - \Delta\rho) v_n(k + \Delta(f(k) - \delta k - c))\}.$$

$$0 = \max_c \{u(c) + \frac{v_n(k + \Delta(f(k) - \delta k - c)) - v_{n+1}(k)}{\Delta} - \rho v_n(k + \Delta(f(k) - \delta k - c))\}.$$

Taking limits and rearranging

$$\rho v_n(k) = \max_c \{u(c) + \lim_{\Delta \to 0} \frac{v_n(k + \Delta(f(k) - \delta k - c)) - v_{n+1}(k)}{\Delta}\}.$$

Why is the contraction property lost?

- Problem

$$\lim_{\Delta \to 0} \frac{v_n(k + \Delta(f(k) - \delta k - c)) - v_{n+1}(k)}{\Delta}$$
$$\neq v_n'(k)(f(k) - \delta - c)$$

- The right hand side of the HJB equation contains v_{n+1} and that's a major issue

How to get it back

- Back to discrete time

$$v_{n+1}(k) = \max_c \{u(c) + (1-\rho)v_n(f(k) + (1-\delta)k - c)\},$$

- Call the optimal choice c_n (it's really a function of k but I'm saving some space)
- Howard's Improvement Algorithm says that we can then iterate on

$$v_{n+1}^{h+1}(k) = u(c_n) + (1-\rho)v_{n+1}^{h}(f(k) + (1-\delta)k - c_n)\},$$

with $v_{n+1}^0 = v_n$.

- Until $v_{n+1}^{h+1} \approx v_{n+1}^h$. This can speed things up considerably, and preserves the contraction property

How to get it back

- Suppose that it holds exactly $v_{n+1}^{h+1} = v_{n+1}^{h}$, and let's just call this function v_{n+1}. Then it must satisfy

$$v_{n+1}(k) = \max_{c}\{u(c_n)+(1-\rho)v_{n+1}(f(k)+(1-\delta)k-c_n)\},$$

- In Δ units of time

$$v_{n+1}(k) = \Delta u(c_n) + (1-\Delta\rho)v_{n+1}(k+\Delta(f(k)-\delta k-c_n)).$$

How to get it back

Rearrange

$$0 = u(c_n) + \frac{v_{n+1}(k + \Delta(f(k) - \delta k - c_n)) - v_{n+1}(k)}{\Delta} \\ - \rho v_{n+1}(k + \Delta(f(k) - \delta k - c_n))\}.$$

and take limits

$$\rho v_{n+1}(k) = u(c_n) + v'_{n+1}(k)(f(k) - \delta k - c_n),$$

How to get it back

$$\rho v_{n+1}(k) = u(c_n) + v'_{n+1}(k)(f(k) - \delta k - c_n),$$

- Now the awkward discrepancy between v_{n+1} and v_n is gone!
- But the problem looks a bit hard to solve!
- Turns out it is not!
- This is where the "implicit method" comes in.

The implicit method

1. Start with a grid for capital $\mathbf{k} = [k_1, k_2, \ldots, k_N]$.
2. For each grid point for capital you have a guess for $v_0(k_i)$, $\forall k_i \in \mathbf{k}$
3. So you have a vector of N values of v_0. Call this $\mathbf{v}_0$
4. You should also have a difference operator (an $N \times N$ matrix) $\mathbf{D}$ such that

$$\mathbf{Dv} \approx \mathbf{v}'(k), \quad \forall k_i \in \mathbf{K}$$

The implicit method

5. Optimal consumption choice given by FOC

$$u'(\mathbf{c}_0) = \mathbf{D}\mathbf{v}_0$$

   reasonable to call this $c(\mathbf{v}_0)$ – an $N \times 1$ vector

6. This implies another $N \times 1$ vector of savings

$$\mathbf{s}_0 = (f(\mathbf{k}) - \delta\mathbf{k} - c(\mathbf{v}_0))$$

   (This vector can be used to improve on $\mathbf{D}$ – more on that in a second).

7. Create the $N \times N$ matrix $\mathbf{S}_0 = diag(\mathbf{s_0})$

The implicit method

That is

$$\mathbf{S} = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & s_N \end{pmatrix},$$

The implicit method

8. Then our HJB equation can now be written as

$$\rho \mathbf{v}_1 = u(c(\mathbf{v}_0)) + \mathbf{S}_0 \mathbf{D} \mathbf{v}_1$$

9. Manipulate

$$(\rho \mathbf{I} - \mathbf{S}_0 \mathbf{D}) \mathbf{v}_1 = u(c(\mathbf{v}_0))$$

10. Lastly

$$\mathbf{v}_1 = (\rho \mathbf{I} - \mathbf{S}_0 \mathbf{D})^{-1} u(c(\mathbf{v}_0))$$

11. Generally

$$\mathbf{v}_{n+1} = (\rho \mathbf{I} - \mathbf{S}_n \mathbf{D})^{-1} u(c(\mathbf{v}_n))$$

The implicit method

Or even more generally

$$\mathbf{v}_{n+1} = ((\rho + 1/\Gamma)\mathbf{I} - \mathbf{S}_n\mathbf{D})^{-1}[u(c(\mathbf{v}_n)) + \mathbf{v}_n/\Gamma]$$

for Γ very large (my experience: $\Gamma = \infty$ is fastest, but set lower if convergence issues arise)

- In matlab always use backslash operator to calculate $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. I.e. $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$

We are talking very substantial speed/robustness gains here. Perhaps by a factor of 1,000.

The implicit method: Improvement trick I

- Yesterday we created the matrix $\mathbf{D}$ as central differences
- We can do better. In particular, $\mathbf{s}_n$ tells us where the economy is drifting for each $k_i \in \mathbf{k}$
- So trick one is to use forward differences for all

$$\{k_i \in \mathbf{k} : s_i > 0\}$$

- and backward differences for all

$$\{k_i \in \mathbf{k} : s_i < 0\}$$

- This leads to

$$\mathbf{v}_{n+1} = ((\rho + 1/\Gamma)\mathbf{I} - \mathbf{S}_n\mathbf{D}_n)^{-1}[u(c(\mathbf{v}_n)) + \mathbf{v}_n/\Gamma]$$

The implicit method: Improvement trick II

- Inspect the matrix

$$((\rho + 1/\Gamma)\mathbf{I} - \mathbf{S}_n\mathbf{D}_n),$$

  and notice that all matrices are super sparse!
- So declaring them as sparse will free up a lot of memory and give you enormous speed gains too (this is particularly true for problems with $N > 200$ or so. Below that it doesn't really matter).
- **Never** declare any of these matrices as anything else than sparse! Use commands as `speye` and `spdiags`
- **Don't** be too concerned about loops. That doesn't seem to be what can clog these systems.

The Aiyagari Model in Continuous Time

- The rest of today's lecture will be to apply our knowledge thus far to the Aiyagari model in continuous time.
- This will also be today's exercise

1. Households' problem
2. Firms problem
3. Equilibrium

Households

- Households can be employed or unemployed
- When employed they receive income $w_t(1-\tau_t)$
- When unemployed they receive unemployment benefits equal to μw_t
- An employed individual becomes unemployed with probability λ_e.
- An unemployed individual becomes employed with probability λ_u
- In an Aiyagari model prices are constant: $r_t = r$ and $w_t = w \ \forall t$

Households

- Dynamics of aggregate unemployment

$$
\begin{aligned}
e_{t+1} &= (1-\lambda_e)e_t + \lambda_u u_t \\
u_{t+1} &= \lambda_e e_t + (1-\lambda_u)u_t
\end{aligned}
$$

- Δ units of time

$$
\begin{aligned}
e_{t+\Delta} &= (1-\Delta\lambda_e)e_t + \Delta\lambda_u u_t \\
u_{t+\Delta} &= \Delta\lambda_e e_t + (1-\Delta\lambda_u)u_t
\end{aligned}
$$

- Rearrange and take limits

$$
\begin{aligned}
\dot{e}_t &= -\lambda_e e_t + \lambda_u u_t \\
\dot{u}_t &= \lambda_e e_t - \lambda_u u_t
\end{aligned}
$$

Households

- System

$$\dot{\mathbf{s}}_t = \mathbf{T}\mathbf{s}_t$$

  with

$$\mathbf{T} = \begin{pmatrix} -\lambda_e & \lambda_u \\ \lambda_e & -\lambda_u \end{pmatrix}$$

- Stationary equilibrium

$$\mathbf{0} = \mathbf{T}\mathbf{s}$$

- Thus **s** is an eigenvector associated with a zero eigenvalue, with the eigenvector normalised to sum to one.

Households

- Can be solved as a regular eigenvalue problem
- But since the eigenvector is only defined up to a scalar we can use the following trick
  1. Create vector

$$\mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and matrix} \quad \hat{\mathbf{T}} = \begin{pmatrix} 1 & 0 \\ \lambda_e & -\lambda_u \end{pmatrix}$$

  2. Find $\hat{\mathbf{s}}$ as $\hat{\mathbf{s}} = \hat{\mathbf{T}}^{-1}\mathbf{b}$.
  3. Normalise $\hat{\mathbf{s}}$ to sum to one to find $\mathbf{s}$.
- The first element of $\mathbf{s}$ is then the stationary employment rate, and the second the stationary unemployment rate.

Households

- Government runs a balanced budget, so not deficits
- The tax rate then solves $u\mu w = e\tau w$
- Or just $\tau = \frac{u}{e}\mu$

Households

- Bellman equation for an employed agent

$$v(a_t, e) = \max_{c_t}\{u(c_t) + (1-\rho)\times \\ [(1-\lambda_e)v(w_t(1-\tau_t) + (1+r_t)a_t - c_t, e) \\ + \lambda_e v(w_t(1-\tau_t) + (1+r_t)a_t - c_t, u)]\}$$

subject to $a_t \geq \phi \ \forall t$.

- Δ units of time

$$v(a_t, e) = \max_{c_t}\{\Delta u(c_t) + (1-\Delta\rho)\times \\ [(1-\Delta\lambda_e)v(\Delta(w_t(1-\tau_t) + r_t a_t - c_t) + a_t, e) \\ + \Delta\lambda_e v(\Delta(w_t(1-\tau_t) + r_t a_t - c_t) + a_t, u)]\}$$

Households

- Rearrange and divide by Δ

$$
\begin{aligned}
0 = \max_{c_t}\{ & u(c_t) \\
& + \frac{v(\Delta(w_t(1-\tau_t)+r_t a_t - c_t)+a_t, e) - v(a_t, e)}{\Delta} \\
& - (\rho + \lambda_e + \Delta\rho\lambda_e)v(\Delta(w_t(1-\tau_t)+r_t a_t - c_t)+a_t, e) \\
& + \lambda_e v(\Delta(w_t(1-\tau_t)+r_t a_t - c_t)+a_t, u)]\}
\end{aligned}
$$

- Take limits and rearrange

$$
\begin{aligned}
\rho v(a,e) = \max_{c}\{ & u(c) + v_a(a,e)(w(1-\tau)+ra-c) \\
& - \lambda_e(v(a,e)-v(a,u))\}
\end{aligned}
$$

Households

So households' problem is given by the two HJB equations

$$\rho v(a,e) = \max_c \{u(c) + v_a(a,e)(w(1-\tau) + ra - c) \\ - \lambda_e(v(a,e) - v(a,u))\}$$

$$\rho v(a,u) = \max_c \{u(c) + v_a(a,u)(w\mu + ra - c) \\ - \lambda_u(v(a,u) - v(a,e))\}$$

Let's be smart in solving them!

Households

1. Start with a linearly spaced grid for assets $\mathbf{a} = [a_1, a_2, \ldots, a_N]$. Let $da = a(n+1) - a(n)$.
2. For each grid for assets guess a for $v_0(a_i, j)$, $\forall a_i \in \mathbf{a}$, and $j \in \{e, u\}$. This gives us $\mathbf{v}_{0,e}$ and $\mathbf{v}_{0,u}$
3. Call the stacked $2N \times 1$ vector $(\mathbf{v}_{0,e}, \mathbf{v}_{0,u})'$ for $\mathbf{v}_0$.

Households

4. Create two $N \times N$ difference operators as

$$\mathbf{D_f} = \begin{pmatrix} -1/da & 1/da & 0 & \dots & 0 \\ 0 & -1/da & 1/da & 0 & \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \dots & & -1/da & 1/da \\ 0 & \dots & & 0 & -1 \end{pmatrix}$$

$$\mathbf{D_b} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -1/da & 1/da & 0 & & \\ \vdots & -1/da & 1/da & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \dots & & -1/da & 1/da \end{pmatrix}$$

Households

5. Create one $2N \times 2N$ matrix as

$$\mathbf{B} = \begin{pmatrix} -\lambda_e & 0 & \dots & 0 & \lambda_e & 0 & \dots & 0 \\ 0 & -\lambda_e & 0 & \dots & 0 & \lambda_e & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \dots & \ddots & \vdots \\ 0 & 0 & 0 & -\lambda_e & 0 & \dots & \dots & \lambda_e \\ \lambda_u & 0 & \dots & 0 & -\lambda_u & 0 & \dots & 0 \\ 0 & \lambda_u & 0 & \dots & 0 & -\lambda_u & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots & \dots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_u & 0 & 0 & \dots & -\lambda_u \end{pmatrix}$$

will be used later

Households

6. Calculate the derivative of the value functions using both forward and backward differences

$$\mathbf{v}'_f(a,e) = \mathbf{D_f}\mathbf{v}_{0,e}, \quad \mathbf{v}'_b(a,e) = \mathbf{D_b}\mathbf{v}_{0,e},$$
$$\mathbf{v}'_f(a,u) = \mathbf{D_f}\mathbf{v}_{0,u}, \quad \mathbf{v}'_b(a,u) = \mathbf{D_b}\mathbf{v}_{0,u},$$

7. Set the **first** elements of $\mathbf{v}'_b(a,e) = u'(w(1-\tau) + r\phi)$ and $\mathbf{v}'_b(a,u) = u'(w\mu + r\phi)$, and the **last** elements of $\mathbf{v}'_f(a,e) = u'(w(1-\tau) + ra_N)$ and $\mathbf{v}'_f(a,u) = u'(w\mu + ra_N)$
8. Find optimal consumption through

$$u'(\mathbf{c}_{e,f}) = \mathbf{D_f}\mathbf{v}_{0,e}, \quad u'(\mathbf{c}_{e,b}) = \mathbf{D_b}\mathbf{v}_{0,e},$$
$$u'(\mathbf{c}_{u,f}) = \mathbf{D_f}\mathbf{v}_{0,u}, \quad u'(\mathbf{c}_{u,b}) = \mathbf{D_b}\mathbf{v}_{0,u},$$

Households

9. Find optimal savings as

$$\mathbf{s}_{e,f} = w(1-\tau) + r\mathbf{a} - \mathbf{c}_{e,f}, \quad \mathbf{s}_{e,b} = w(1-\tau) + r\mathbf{a} - \mathbf{c}_{e,b},$$
$$\mathbf{s}_{u,f} = w\mu + r\mathbf{a} - \mathbf{c}_{u,f}, \quad \mathbf{s}_{u,b} = w\mu + r\mathbf{a} - \mathbf{c}_{u,b}$$

10. Create indicator vectors

$$\mathbf{I}_{e,f} = (I_{1,e,f}, I_{2,e,f}, \ldots, I_{N,e,f})', \quad \mathbf{I}_{e,b} = (I_{1,e,b}, I_{2,e,f}, \ldots, I_{N,e,b})',$$
$$\mathbf{I}_{u,f} = (I_{1,u,f}, I_{2,u,f}, \ldots, I_{N,u,f})', \quad \mathbf{I}_{u,b} = (I_{1,u,f}, I_{2,u,f}, \ldots, I_{N,u,f})',$$

where $I_{i,j,f} = 1$ **if** $s_{i,j,f} > 0$ and $I_{i,j,b} = 1$ **if** $s_{i,j,b} < 0$, for $i = 1, \ldots, N$ and $j \in \{e, u\}$.

Households

11. Find consumption as

$$\mathbf{c}_e = \mathbf{l}_{e,f} \cdot \mathbf{c}_{e,f} + \mathbf{l}_{e,b} \cdot \mathbf{c}_{e,b}$$
$$\mathbf{c}_u = \mathbf{l}_{u,f} \cdot \mathbf{c}_{u,f} + \mathbf{l}_{u,b} \cdot \mathbf{c}_{u,b}$$

12. Find savings as

$$\mathbf{s}_e = \mathbf{l}_{e,f} \cdot \mathbf{s}_{e,f} + \mathbf{l}_{e,b} \cdot \mathbf{s}_{e,b}$$
$$\mathbf{s}_u = \mathbf{l}_{u,f} \cdot \mathbf{s}_{u,f} + \mathbf{l}_{u,b} \cdot \mathbf{s}_{u,b}$$

13. And matrices $\mathbf{S}_e\mathbf{D}_e$ and $\mathbf{S}_u\mathbf{D}_u$ as

$$\mathbf{S}_e\mathbf{D}_e = diag(\mathbf{l}_{e,f} \cdot \mathbf{s}_{e,f})\mathbf{D_f} + diag(\mathbf{l}_{e,b} \cdot \mathbf{s}_{e,b})\mathbf{D_b}$$
$$\mathbf{S}_u\mathbf{D}_u = diag(\mathbf{l}_{u,f} \cdot \mathbf{s}_{u,f})\mathbf{D_f} + diag(\mathbf{l}_{u,b} \cdot \mathbf{s}_{u,b})\mathbf{D_b}$$

Households

14. Lastly find the $2N \times 2N$ matrix $\mathbf{S}_0\mathbf{D}_0$ as

$$\mathbf{S}_0\mathbf{D}_0 = \begin{pmatrix} \mathbf{S}_e\mathbf{D}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_u\mathbf{D}_u \end{pmatrix}$$

15. And the matrix $\mathbf{P}_0$ as

$$\mathbf{P}_0 = \mathbf{S}_0\mathbf{D}_0 + \mathbf{B}$$

Households

Using the implicit method the households' problem is given by the two HJB equations

$$\begin{aligned}\rho v_{n+1}(a,e) = u(c_n) + v_{a,n+1}(a,e)(w(1-\tau) + ra - c_n) \\ - \lambda_e(v_{n+1}(a,e) - v_{n+1}(a,u))\end{aligned}$$

$$\begin{aligned}\rho v_{n+1}(a,u) = u(c_n) + v_{a,n+1}(a,u)(w\mu + ra - c) \\ - \lambda_u(v_{n+1}(a,u) - v_{n+1}(a,e))\end{aligned}$$

Households

- These can now be written as

$$\rho \mathbf{v}_{n+1} = u(\mathbf{c}_n) + \mathbf{P}_n \mathbf{v}_{n+1}$$

  with $\mathbf{c}_n = (\mathbf{c}_{n,e}, \mathbf{c}_{n,u})$.
- So we iterate on

$$\mathbf{v}_{n+1} = [(\rho + 1/\Gamma)\mathbf{I} - \mathbf{P}_n]^{-1}[u(\mathbf{c}_n) + \mathbf{v}_n/\Gamma]$$

  until convergence

Firms

- Firms fact the standard static optimisations problem

$$\Pi_t = \max\{K_t^{\alpha} N_t^{1-\alpha} - w_t N_t - (r_t + \delta) K_t\}$$

- With first order conditions

$$r_t = \alpha \left(\frac{K_t}{N_t}\right)^{\alpha-1} - \delta, \quad w_t = (1-\alpha)\left(\frac{K_t}{N_t}\right)^{\alpha}$$

- In a stationary equilibrium this implies

$$r = \alpha \left(\frac{K}{(1-u)}\right)^{\alpha-1} - \delta, \quad w = (1-\alpha)\left(\frac{r+\delta}{\alpha}\right)^{\frac{\alpha}{\alpha-1}}$$

Stationary distribution

- What is the evolution of the endogenous stationary distribution of wealth and employment status?
- Denote the CDF as $G_{t+1}(a, e)$. This must satisfy

$$G_{t+1}(a, e) = (1 - \lambda_e) G_t(a^e_{-1}, e) + \lambda_u G_t(a^u_{-1}, u),$$

  where a^j_{-1} denotes "where you came from" from optimally setting $a_{t+1} = a$ in employment status $j \in \{e, u\}$.
- In Δ units of time approximate this as $a^e_{-1} = a - \Delta s_e$ and $a - \Delta s_u$. Thus

$$G_{t+\Delta}(a, e) = (1 - \Delta\lambda_e) G_t(a - \Delta s_e, e) + \Delta\lambda_u G_t(a - \Delta s_u, u),$$

Stationary distribution

$$G_{t+\Delta}(a,e) = (1-\Delta\lambda_e)G_t(a-\Delta s_e, e) + \Delta\lambda_u G_t(a-\Delta s_u, u),$$

- Subtract $G_t(a,e)$ from both sides and divide by Δ

$$\frac{G_{t+\Delta}(a,e) - G_t(a,e)}{\Delta} = \frac{G_t(a-\Delta s_e, e) - G_t(a,e)}{\Delta} - \lambda_e G_t(a-\Delta s_e, e) + \lambda_u G_t(a-\Delta s_u, u),$$

- Take limits

$$\dot{G}_t(a,e) = -g_t(a,e)s_e(a) - \lambda_e G_t(a,e) + \lambda_u G_t(a,u),$$

Stationary distribution/Kolmogorov Forward Equation

$$\dot{G}_t(a,e) = -g_t(a,e)s_e(a) - \lambda_e G_t(a,e) + \lambda_u G_t(a,u),$$

- Differentiate with respect to a

$$\dot{g}_t(a,e) = -\frac{\partial[g_t(a,e)s_e(a)]}{\partial a} - \lambda_e g_t(a,e) + \lambda_u g_t(a,u),$$

- Thus the law of motion for the endogenous distribution is

$$\dot{g}_t(a,e) = -\frac{\partial[g_t(a,e)s_e(a)]}{\partial a} - \lambda_e g_t(a,e) + \lambda_u g_t(a,u),$$
$$\dot{g}_t(a,u) = -\frac{\partial[g_t(a,u)s_u(a)]}{\partial a} - \lambda_u g_t(a,u) + \lambda_e g_t(a,e)$$

Stationary distribution/Kolmogorov Forward Equation

- Remember the matrix

$$\mathbf{P}_n = \mathbf{S}_n \mathbf{D}_n + \mathbf{B}.$$

- When converged

$$\mathbf{P} = \mathbf{SD} + \mathbf{B}$$

- Turns out that

$$\dot{\mathbf{g}}_t = \mathbf{P}' \mathbf{g}_t$$

- Where $\mathbf{g}_t$ is the stacked vector $(\mathbf{g}_t(a, e), \mathbf{g}_t(a, u))'$

Solving the Aiyagari model

1. Guess for an interest rate r_n. Find w_n as

$$w_n = (1-\alpha)\left(\frac{r_n+\delta}{\alpha}\right)^{\frac{\alpha}{\alpha-1}}$$

2. Find $\mathbf{v}$ such that

$$\mathbf{v} = [(\rho + 1/\Gamma)\mathbf{I} - \mathbf{P}]^{-1}[u(c(\mathbf{v})) + \mathbf{v}/\Gamma]$$

3. Find $\mathbf{g}$ by solving

$$\mathbf{0} = \mathbf{P}'\mathbf{g}$$

   and normalise to sum to one (remember how we found $\mathbf{s}$ above)

Solving the Aiyagari model

4. Find K_n as

$$K_n = \mathbf{g}' \begin{pmatrix} \mathbf{a} \\ \mathbf{a} \end{pmatrix}$$

5. Find $\hat{r}$ as

$$\hat{r} = \alpha \left(\frac{K_n}{(1-u)} \right)^{\alpha-1} - \delta$$

6. If $\hat{r} > r$ set $r_{n+1} > r_n$, else set $r_{n+1} < r_n$.
7. Repeat until $\hat{r} \approx r_n$.