Solving Heterogeneous Agent Macro Models in Continuous Time

Eirik Eylands Brandsås*

December 12, 2019

1 Introduction

In this lecture note, we will write down a continuous time variant of the Aiyagari-Bewley-Hugget type models, solve it using finite difference upwinding and extend the model to include housing. The two main goals of the lecture is to a) teach you another useful computational tool and b) to increase your exposure to continuous time methods in a familiar setting (Heterogeneous Agents)). The notes relies very heavily on Achdou et al. (2017a) and its numerical appendix (Achdou et al., 2017b). Candler (1999) is another potentially useful reference.

Code is published in a Github repository: `ContTimeHetAgentExample`. Currently the repo(sitory) contains code in the following languages:

Model	Matlab	Julia
Benchmark	Y	Y
Extension with housing	Y	N

1.1 Preview of Results

At the end of this lecture, you will be able to 'instantly' solve equilibrium models with heterogeneous agents, and find the joint income-wealth distributions. In addition, we will deal with non-convexities, models with multiple stationary distributions and financial frictions in a model with housing and a downpayment requirement.

*University of Wisconsin-Madison. E-mail: brandsaas@wisc.edu.

2 Benchmark Model

Households maximize life time expected utility

$$\mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t)\, \mathrm{d}t, \tag{1}$$

$$\dot{a}_t = y_t + r_t a_t - c_t, \tag{2}$$

$$a_t \geq 0. \tag{3}$$

where $\dot{a}$ denotes the growth of assets, y is income, r is the interest rate and c is the consumption choice. Note that $y, c, \dot{a}$ are flows while a is a stock. The income process y_t is an endowment that follows a Poisson process ($\approx$ continuous time Markov Chain) where λ_{ij} denote the arrival rate of shocks moving the household from income y_i to y_j. For simplicity, assume there are only two income states, high and low. Note how similar this is to what you've seen before in 712 and in this class. See section 2.1 for how to derive the budget constraint.

As in discrete time we want to rewrite this problem into a recursive form, here called the Hamilton–Jacobi–Bellman (HJB) equation. There are (at least) three methods to derive this equation, and all yield the same answer: 1) writing down the problem in discrete time and letting the time step go to zero, 2) invoking Ito's Lemma for jump processes or 3) a second order Taylor expansion. See appendix A for a sketch of how to do it by letting the time-step go to zero. Barczyk and Kredler (2017) show how to do it by a Taylor expansion. *This is the HJB equation, no matter how you derive it:*

$$\rho V(a,y) = \max_c \left\{ u(c) + \dot{a} V_a(a,y) \right\} + \lambda [V(a,\tilde{y}) - V(a,y)], \tag{4}$$

$$\text{s.t:} \quad \dot{a} = ar - c + y. \tag{5}$$

Here V is the value function, $V_a(\cdot)$ denotes the derivative of V with respect to its first argument a, λ are the arrival rate of income changes ($\approx$ jump probabilities), and $\tilde{y}$ denotes the other income value.

2.1 Budget Constraint

In discrete time we have:

$$a_{t+1} = a_t(1+r) + y_t - c_t.$$

Change the time step from 1 to Δ:

$$a_{t+\Delta} = a_t(1 + r\Delta) + y_t\Delta - c_t\Delta,$$

note how in discrete time we implicitly multiply consumption of each period with the time-step that is 1-period. Now, divide through by Δ and let it go to zero:

$$\frac{a_{t+\Delta} - a_t}{\Delta} = a_t r + y_t - c_t$$
$$\implies \lim_{\Delta \to 0} \frac{a_{t+\Delta} - a_t}{\Delta} = \frac{\mathrm{d}a}{\mathrm{d}t} \equiv \dot{a} = ar - y + y$$

2.2 First Order Condition

Note that the problem in (4) can be solved easily since u is concave, $V_a(a, y)$ is constant for (a, y) and positive, so we have an interior solution:[1]

$$u_c(c) = V_a(a, y) \implies c = (u_c)^{-1} (V_a(a, y)) . \tag{6}$$

Note that relative to working in discrete time the main two simplifications already are that 1) The borrowing constraint 'disappears', see section 3.3 and 2) that we can solve for c as a function of V, *without* expectations. In discrete time the corresponding FOC would be of the form $c = (u_c^{-1} (\beta(1 + r)\mathbb{E}[V_a(a', y')])$ which we for general income processes may not have closed form solution for and potentially with a Kuhn-Tucker multiplier for the borrowing constraint.

2.3 Continuous Time Technicalities - Classical vs Weak Solutions

A classical solution to a Partial Differential Equation like (4) is continuously differentiable as many times as needed, here one time. However, with borrowing constraints or financial frictions, we expect the value function to have kinks. For that reason, we use what is called weak solutions, which need not be continuously differentiable, but must satisfy the equation in 'some way'. In this setup, this means that we find a viscosity solution for the HJB equation and a measure-valued solution for the so-called Kolmogorov Forward-equation (23) (which describes the evolution of the measure of households).

Now, what do we do when $\dot{a}_{i,j,F} > 0$ *and* $\dot{a}_{i,j,B} < 0$? We first define an indicator $\mathbf{1}_{i,j}^{unique}$ for when the drifts 'agree', and when for when they disagree $\mathbf{1}_{i,j}^{both}$. Then, when the drifts disagree, we use whichever yields the largest Hamiltonian.

3 Finite differences and upwinding

Let $\partial V_{i,j}$ denote the numerical partial derivative of V at grid a_i, y_j with respect to assets. We can write out the discretized value function as

$$\rho V_{i,j} = u(c) + \dot{a}\partial V_{i,j} + \lambda[V_{i,-j} - V_{i,j}] \tag{7}$$

[1] Question: What happens if $V_a <= 0$?

3.1 Finite difference

We need to find numerical derivatives for the value function, which can be done in many ways. The arguably best way is to use finite differences. Let Δa define the distance between two grid points on the asset grid, e.g. $\Delta a = a_i - a_{i-1}$. The backward and forward (first) derivatives are defined as

$$\partial V_{i,j,F} = \frac{V_{i,j} - V_{i-1,j}}{\Delta a}, \tag{8}$$

$$\partial V_{i,j,B} = \frac{V_{i+1,j} - V_{i,j}}{\Delta a}. \tag{9}$$

A method that is easier to implement is to always use the centered derivative and ignore upwinding. The centered derivative is

$$\partial V_{i,j,C} = \frac{V_{i+1,j} - V_{i-1,j}}{\Delta a}.$$

However, this method does not work well in practice and will often break. My intuition is that the upwind method is preferred because it uses the forward derivative when we are drifting up, so that it takes into account the 'dynamic' nature of the drift, and uses a more accurate derivative for the given movement.

3.2 Upwinding (for concave functions)

The next question is to decide when do we use what derivative? Upwinding tells us to use the up-derivative when that variable is drifting up and vice versa. You may the Wikipedia article on upwinding helpful: `https://en.wikipedia.org/wiki/Upwind_scheme`. First, find back/forward savings by:

$$\begin{aligned} \dot{a}_{i,j,F} &= y_j + ra_i - (u_c)^{-1}(\partial V_{i,j,F}) \\ \dot{a}_{i,j,B} &= y_j + ra_i - (u_c)^{-1}(\partial V_{i,j,B}), \end{aligned} \tag{10}$$

which also define $c^F_{i,j}, c^B_{i,j}$.

Use the following rule for determining which derivative to use:

$$\partial V_{i,j} = \partial_F V_{i,j} \mathbf{1}_{\{\dot{a}_{i,j,F} > 0\}} + \partial_B V_{i,j} \mathbf{1}_{\{\dot{a}_{i,j,B} < 0\}} + \partial V_{i,j,0} \mathbf{1}_{\{\dot{a}_{i,j,F} \leq 0 \leq \dot{a}_{i,j,B}\}}, \tag{11}$$

where, when the forward/back drift disagrees on the sign, we use the 'staying put' derivative $\partial V_{i,j,0}$. First, note that since V is concave $\partial V_{i,j,F} < \partial V_{i,j,B}$. When the tiebreaker rule is in effect we set $\dot{a}_{i,j,0} = 0$, and thenfind the derivative of the value function from the Envelope theorem:

$$\dot{a}_{i,j} = 0 \implies c_{i,j} = ra_i + y_j \implies \partial V_{i,j,0} \equiv u_c(ra_i + y_j) \tag{12}$$

3.3 Lets talk about the borrowing constraint

So far we have ignored the borrowing constraint completely. We need some restriction so that the solution satisfies the borrowing constraint. We can derive the necessary condition on the value function:

$$\partial V_{1,j,B} = u_c(y_j + ra_1)$$

Note that from (11) we can see that this condition will only be used when $\dot{a}_{1,j,F} < 0$ If the forward policy rule implied positive drift at the boundary this state constraint does not bite, since $\dot{a}_{i,j,F} < \dot{a}_{i,j,B}$.

Another way is to 'brute force' it, by checking (for both f, b) :

$$c^f_{1,j} = \max\{y_j + ra_1, \tilde{c}^f\},$$

where $\tilde{c}$ denotes the derivative before enforcing any constraints. If you use the latter approach it is important to enforce it before you find the indicators in (11). The advantage of the latter method is that it can be easier to implement with many choices.

4 Finding the Value Function

4.1 Intro

We have now found all policy functions given a value function V^n, and the remaining step is to update the value function. You may the Wikipedia article on Finite Differences helpful: `https://en.wikipedia.org/wiki/Finite_difference_method`. Section 4.2 contains one of the key benefits of the finite-difference method: we can write out the value function updating equation as a linear system which we can solve extremely fast using standard techniques.

We will do so by the implicit method, but the most 'natural' way is the explicit method:

4.1.1 Explicit method

$$\frac{V^{n+1}_{i,j} - V^n_{i,j}}{\Delta} + \rho V^n i, j = u(c^n_{i,j}) + \dot{a}\partial V^n_{i,j} + \lambda[V^n_{i,-j} - V^n_{i,j}], \tag{13}$$

where Δ is the step length, and so here the n superscript essentially denotes time instead of iterations, and we iterate until the time derivative is zero (since the model is stationary we know that $\frac{V^{n+1}_{i,j} - V^n_{i,j}}{\Delta} \approx \frac{\partial V}{\partial t} \approx= 0$). Note that $c^n, \partial V^n, \dot{a}^n$ are all found using upwinding. Also note how similar this is to standard value function iteration.

1. Start with an initial guess

2. Given the guess, find forward/back derivatives

3. Given the derivatives, find the implied drifts $\dot{a}$

4. Given the drifts, use the upwinding rule in (11) to find the indicators

5. Use the indicators to find which drift and consumption to use

6. Find the new value function from equation (13)

7. Repeat steps 2-6 until the value function converges

Note that the explicit method requires small time steps (the maximum size is given by the so-called Courant-Friedrichs-Lewy (CFL) condition) and is not robust.

4.2 Implicit Method

However, the best way to update the value function is to use the implicit method, where we use the following updating rule:

$$\frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta} + \rho V_{i,j}^{n+1} = u(c_{i,j}^{n}) + \dot{a}_{i,j}^{n} \partial V_{i,j}^{n+1} + \lambda[V_{i,-j}^{n+1} - V_{i,j}^{n+1}], \tag{14}$$

where all policies are denoted n and all value functions are denoted $n+1$. We can write this out by inserting for upwinding:

$$\frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta} + \rho V_{i,j}^{n+1} = u(c_{i,j}^{n}) + \mathbf{1}_F \dot{a}_{i,j,F} \partial V_{i,j,F}^{n+1} + \mathbf{1}_B \dot{a}_{i,j,B} \partial V_{i,j,B}^{n+1} + \lambda[V_{i,-j}^{n+1} - V_{i,j}^{n+1}], \tag{15}$$

where $\mathbf{1}_F \equiv \mathbf{1}_{\{ra_i + y_j - c_{i,j,F} > 0\}}$, and $c_{i,j}^{n}$ is still found by upwinding. The key is to recognize that this is a linear system: We have N grid points for assets and 2 for income, and so this is a system of $2 \times N$ linear equations. This is becomes clear when we insert also for the derivatives:

$$\begin{aligned} \frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta} + \rho V_{i,j}^{n+1} =& u(c_{i,j}^{n}) + \mathbf{1}_F \dot{a}_{i,j,F} \frac{V_{i+1,j}^{n+1} - V_{i,j}^{n+1}}{\Delta a} + \mathbf{1}_B \dot{a}_{i,j,B} \frac{V_{i,j}^{n+1} - V_{i-1,j}^{n+1}}{\Delta a} \\ &+ \lambda[V_{i,-j}^{n+1} - V_{i,j}^{n+1}], \end{aligned} \tag{16}$$

note that all terms on the right-hand side are evaluated at grid point $i+1, i, i-1$, so we can collect all 'weights' into x, y, z (back, stay, forward) depending on whether they load on V_{i-1}, V_i, V_{i+1}:

$$\frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta} + \rho V_{i,j}^{n+1} = u(c_{i,j}^{n}) + V_{i-1,j}^{n+1} x_{i,j} + V_{i,j}^{n+1} y_{i,j} + V_{i+1,j}^{n+1} z_{i,j} + \lambda V_{i,-j}^{n+1}, \tag{17}$$

where each weight is given by

$$x_{i,j} = -\frac{\mathbf{1}_B \dot{a}_{i,j,B}}{\Delta a}.$$
$$y_{i,j} = \frac{\mathbf{1}_B \dot{a}_{i,j,B}}{\Delta a} - \frac{\mathbf{1}_F \dot{a}_{i,h,F}}{\Delta a} - \lambda.$$
$$z_{i,j} = \frac{\mathbf{1}_F \dot{a}_{i,j,F}}{\Delta a}.$$

Note importantly that $x_{1,j} = z_{N,j} = 0$, so that $V_{0,1}, V_{N+1,j}$ are never used. Equation (17) can be written as:

$$\frac{1}{\Delta}(V^{n+1} - V^n) + \rho V^{n+1} = u^n + \mathbf{A}^n V^{n+1}, \tag{18}$$

where we have stacked the value functions and the utility functions into vectors:

$$V^n = \begin{bmatrix} V^n_{1,1} \\ \vdots \\ V^n_{N,1} \\ V^n_{1,2} \\ \vdots \\ V^n_{N,2} \end{bmatrix}, \text{ and } u^n = \begin{bmatrix} u(c^n_{1,1}) \\ \vdots \\ u(c^n_{N,1}) \\ u(c^n_{1,2}) \\ \vdots \\ u(c^n_{N,2}) \end{bmatrix}$$

where the matrix $\mathbf{A}^n$ is given by:

$$\mathbf{A}^n = \left[\begin{array}{ccccc|ccccc} y_{1,1} & z_{1,1} & 0 & \dots & 0 & \lambda & 0 & 0 & \dots & 0 \\ x_{2,1} & y_{2,1} & z_{2,1} & 0 & 0 & 0 & \lambda & 0 & 0 & 0 \\ 0 & x_{3,1} & y_{3,1} & z_{3,1} & 0 & 0 & 0 & \lambda & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \lambda & \vdots \\ 0 & \dots & \dots & x_{na,1} & y_{na,1} & 0 & \dots & \dots & 0 & \lambda \\ \hline \lambda & 0 & 0 & \dots & 0 & y_{1,2} & z_{1,2} & 0 & \dots & 0 \\ 0 & \lambda & 0 & 0 & 0 & x_{2,2} & y_{2,2} & z_{2,2} & 0 & \dots \\ 0 & 0 & \lambda & 0 & 0 & 0 & x_{3,2} & y_{3,2} & z_{3,2} & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & \lambda & 0 & \dots & \dots & x_{na,2} & y_{na,2} \end{array}\right] \tag{19}$$

Note that this matrix is block diagonal and extremely sparse, each row has at most 4 non-zero elements.

We can write the system as

$$\mathbf{B}^n V^{n+1} = b^n, \quad \mathbf{B}^n = \left(\frac{1}{\Delta} + \rho\right)\mathbf{I} - \mathbf{A}^n, \quad b^n = u^n + \frac{1}{\Delta}V^n$$

and thus we have have a closed form solution for the updated value function at every grid point

$$V^{n+1} = (\mathbf{B^n})^{-1} b^n \tag{20}$$

4.2.1 Summary of Algorithm

The algorithm is identical to the one of the explicit method, just that we update the value function from equation (20) instead of using equation (13).

4.3 Sidenote

Note that if $\Delta = \infty$, we get:

$$\rho V^{n+1} = u^n + A^n V^{n+1}, \tag{21}$$

in other words, we have essentially just rewritten the HJB equation into a linear system where A^n denotes transition probabilities as a function of the household's choices. One can then think of A^n as a standard transition matrix, and one can see that each row of A^n sums to zero, diagonal elements are non-positive and off-diagonal elements are non-negative. If all elements in a row were zero, that state would be absorbing.

5 Finding the Distribution

Let $g_{i,j}$ denote the mass of households with assets a_i and income y_j, so that we have

$$\int_0^\infty g(a, y_1)\,\mathrm{d}a + \int_0^\infty g(a, y_2)\,\mathrm{d}a = 1 \tag{22}$$

The (stationary) equilibrium condition is called the Kolmogorov Forward (KF) or Fokker-Planck equation:

$$0 = -\frac{\mathrm{d}}{\mathrm{d}a}[\dot{a}(a, y_j)g(a, y_j)] - \lambda g(a, y_j) + \lambda g(a, y_{-j}) \tag{23}$$

which is essentially saying that that the net drift in/out of (a, j) must be zero. How do we find this equation? It turn's out this matrix $\mathbf{A}$ gives us the stationary distribution! Write out the upwind approximation for this:

$$-\frac{\mathbf{1}_F \dot{a}_{i,j,F} g_{i,j} - \mathbf{1}_F \dot{a}_{i-1,j,F} g_{i-1,j}}{\Delta a} - \frac{\mathbf{1}_B \dot{a}_{i+1,j,B} g_{i+1,j} - \mathbf{1}_B \dot{a}_{i,j,B} g_{i,j}}{\Delta a} - g_{i,j}\lambda_j - g_{i,-j}\lambda_j = 0,$$

and note that the forward are now at $i, i-1$ and the back at $i+1, i$ - since we are drifting 'out', while in the value function iteration we were drifting 'into'. We can write this as we did before to a linear system:

$$\begin{aligned}
0 &= g_{i-1,j-1} z_{i-1,j} + g_{i,j} y_{i,j} + g_{i+1,j} x_{i+1,j} + g_{i,-j}\lambda \\
x_{i+1,j} &= -\frac{\mathbf{1}_B \dot{a}_{i+1,j,B}}{\Delta a}. \\
y_{i,j} &= \frac{\mathbf{1}_B \dot{a}_{i,j,B}}{\Delta a} - \frac{\mathbf{1}_F \dot{a}_{i,h,F}}{\Delta a} - \lambda. \\
z_{i-1,j} &= \frac{\mathbf{1}_F \dot{a}_{i-1,j,F}}{\Delta a}.
\end{aligned}$$

From these expressions we see clearly that we are capturing the mass that flows up from $i-1, j$, the mass that stays, the mass that flows down from $i+1, j$ and the mass that jumps from $i, -j$. But then we see that this defines the following equation:

$$\mathbf{A}^T g = 0, \tag{24}$$

where T denotes the transpose. In the language of dynamic control, $\mathbf{A}$ is the discretized infintesimal generator and $\mathbf{A}^T$ is the discretized Kolmogorov Forward operator.

5.1 Note

Note that equation (24) is an eigenvalue problem. The matrix $\mathbf{A}^T$ is singular, so to invert you either re-normalize one row (see the Matlab code for an example) or you use an eigenvalue solver. The key is that any non-negative vector g that solves equation 24 is a stationary distribution, and there are many ways to find that equation.

5.2 Equilibrium

The last remaining piece is to close the asset market. The easiest way is to use Hugget's version, where the net supply of bonds equals some exogenous level $\bar{b}$, say $\bar{b} = 0$:

$$\begin{aligned} 0 &= \int_0^\infty a g(a,1)\,\mathrm{d}a + \int_0^\infty a g(a,2)\,\mathrm{d}a = A(g;r) \\ \implies 0 &= \sum_{i=1}^N a_i g_{i,1} \Delta a + \sum_{i=1}^N a_i g_{i,2} \Delta a \end{aligned}$$

To be more formal we are looking for a solution $\mathbf{V}, \mathbf{g} S(\mathbf{g}, r)$ such that the following equations hold:

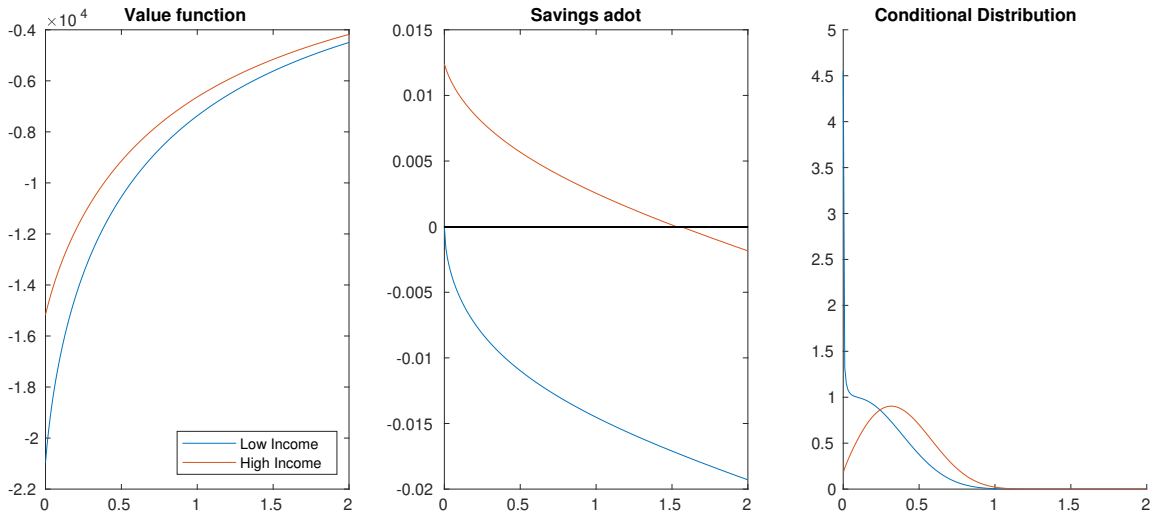
$$\begin{aligned} \rho \mathbf{V} &= u(\mathbf{c}) + \mathbf{A}(\mathbf{V}; r)\mathbf{V} \\ 0 &= \mathbf{A}(\mathbf{V}; r)^T \mathbf{g} \\ \bar{b} &= S(\mathbf{g}; r) \end{aligned} \tag{25}$$

where the equations are the discretized (stacked) value function, the discretized (stacked) KF equation and the market clearing condition.

5.3 Model Solution

We can then finally plot the decision rules that the model spits out, and find the stationary distribution for a given level of prices, see figure 1. Note how the policy function for savings mimics the policy rule in discrete time, just that the borrowing constraint only binds for a broke household with low income.

Figure 1: Solution to the Standard Heterogeneous Agent Model



6 Benefits and Costs of Continuous Time

As I see it there are four major benefits of this framework:

- The HJB is in some sense static, leading to very easy solutions for the policy functions.
- The borrowing constraint never binds in the interior of the state space, so we can ignore problems associated with occasionally binding constraints.
- To find the HJB equation we also find the transition matrix of the economy, so we finding the distribution for any set of parameter values is 'free'.
- The method is extremely fast due to the pattern of the matrix $\mathbf{A}$: It is sparse and (block) diagonal, which means that it can be solved efficiently using standard techniques implemented naturally in most programming languages
- The method is extremely easy to extend. One can include portfolio choices, housing, multiple generations, life-cycle, labor-leisure choices all with minimal modification to the solution algorithm

There is no free lunch, however:

- The method is generally not as robust as the standard discrete time value function iteration. In particular, it struggles when there is a lot of curvature and there are some issues regarding discrete choices ('optimal stopping time').
- The method can struggle as the HJB equation becomes increasingly non-linear, and may require 'too many' grid points

- As you add more states you get more drift terms, and so the **A** becomes less and less block-diagonal, which reduces the speed of inverting it.

7 Example of Extension: Housing

It is often said that one loses many of the benefits from continuous time models when the model is no longer 'smooth'. As a counterexample, we show how to solve the model with indivisible housing and a down-payment constraint, so that we include financial frictions and indivisible assets. Sounds complicated, but it's straightforward. For simplicity, we just solve the decision problem and find the distribution.

7.1 Primitives

Households have preferences over consumption and housing services h:

$$\mathbb{E}_0 \int_0^\infty e^{-\rho t} U(c_t, h_t)\, \mathrm{d}t. \tag{26}$$

Households can borrow and save in a risk free bond b_t and buy housing services at a price p. There are no house sizes below $h_{min} > 0$, so a household chooses between not owning a house $h = 0$ or having a house larger than h_{min}. The budget constraint is given by:

$$\dot{b} + p\dot{h} = y + rb - c$$

To buy a house the household must afford the downpayment of d as a fraction of the market value:

$$-b \leq (1-d)ph.$$

We thus have a model with collaterilized lending with a loan-to-value constraint. Assume for simplicity that utility is seperable:

$$U(c, h) = u(c + f(h)) \tag{27}$$

7.2 Writing down the problem

We rewrite the problem to be formulated in net-worth terms: $a = b + ph$, and $\dot{a} = y + r(a - ph) - c$. The borrowing constraint changes to be $ph < \frac{1}{d}a$, and we can denote the set of feasible housing choices by $\mathcal{H}(a)$:

$$\mathcal{H}(a) = \{h : dph \leq a\} \cap \{[0, [h_{min}, \infty]\}$$

We can then write down the HJB equation:

$$\rho V(a, y) = \max_{c, h \in \mathcal{H}, c + f(h) \geq 0} \left\{ u(c + f(h)) + V_a(y + r(a - ph) - c) \right\} - \lambda[V(a, y) - V(a, \tilde{y})] \tag{28}$$

We can solve the model two ways, either by finding the policies numerically, or we can do a change of variables. First, define a function $\tilde{f}(a)$ that denotes the pecuniary equivalent of utility from housing:

$$\tilde{f}(a) = \max_{h \in \mathcal{H}(a)} \{f(h) - rph\}, \tag{29}$$

Then, define aggregate consumption $x = c + f(h)$, and we can rewrite the HJB equation:

$$\begin{aligned}\rho V(a,y) &= \max_{c,h} \left\{\tilde{u}(x) + V_a(y + ra + \tilde{f}(a) - x)\right\} - \lambda[V(a,y) - V(a,\tilde{y})] \\ \tilde{f}(a) &= \max_{h \in \mathcal{H}(a)} \{f(h) - rph\}\end{aligned}$$

Note the particular formulation of utility: We consume housing only until $f'(h) = rp$, after which we increase our goods consumption. If $f'(h_{min}) \geq rp$ we will always consume some housing if it is feasible. Further, the maximizer of $\tilde{f}(a)$ does not depend on the value function V. Note also that by splitting up the problem we've found a very simple way to deal with the borrowing constraint and indivisibility of housing.

7.3 Upwinding - Convex Case

With this specification, we can no longer expect the value function V to be strictly concave, and certainly not continuously differentiable in the interior. First, we here must use weak solutions. Secondly, we expect some convexities and kinks in the value function no longer have that $\dot{a}_{i,j,F} < \dot{a}_{i,j,B}$. As discussed in section 2.3 our solution method is robust to kinks, but we must explicitly take care of convexity of the value function in the upwind scheme

First, define the forward/back Hamiltonians (the parts of the choice problem you can control:

$$H_{i,j,F} = u(x_{i,j,F}) + \partial V_{i,j,F}\dot{a}_{i,j,F}, \quad H_{i,j,B} = u(x_{i,j,B}) + \partial V_{i,j,F}\dot{a}_{i,j,B}, \tag{30}$$

We then define the upwind rule as follows:

$$\begin{aligned}\partial V_{i,j} = &\partial V_{i,j,F}\left(\mathbf{1}_{i,j,F}\mathbf{1}_{i,j}^{unique} + \mathbf{1}_{i,j}^{both}\mathbf{1}_{H_{i,j,F} \geq H_{i,j,B}}\right) \\ &+ \partial V_{i,j,B}\left(\mathbf{1}_{i,j,B}\mathbf{1}_{i,j}^{unique} + \mathbf{1}_{i,j}^{both}\mathbf{1}_{H_{i,j,F} < H_{i,j,B}}\right) \\ &+ \partial V_{i,j,0}\mathbf{1}_{\{\dot{a}_{i,j,F} \leq 0 \leq \dot{a}_{i,j,B}\}}.\end{aligned} \tag{31}$$

Define new indicators $\tilde{\mathbf{1}}_{i,j,F} = \mathbf{1}_{i,j,F}\mathbf{1}_{i,j}^{unique} + \mathbf{1}_{i,j}^{both}\mathbf{1}_{H_{i,j,F} \geq H_{i,j,B}}$ and similarly for $\tilde{\mathbf{1}}_{i,j,B}$ for the implicit method (section 4.2).

7.4 Finding The Distribution - Iterative Approach

In the benchmark model we have a unique stationary equilibrium, so solved directly for the fixed points $g = A^T g$. With non-convex problems we generally don't get unique stationary distributions, so the final distribution depends on where we start.

We can derive the updating equation heuristically, by moving backward in time. If the mass at time $t + \Delta$ is $g_{t+\Delta}$, that must mean that the mass at at t was equal to $g_t = g_{t+\Delta} - A^T \Delta g_{t+\Delta}$, i.e. yesterday's mass is equal's to today's mass minus the flow in today's mass.

$$g_{t+\Delta}(I - A^T\Delta) = g_t \implies g_{t+\Delta} = (I - A^T\Delta)^{-1} g_t \tag{32}$$

We can then start with some initial distribution g_0 and iterate forward in time until we find a fixed point.

7.5 Solution - With Housing

Figure 2 plots the solution. Note that the non-convexity of the maximization problem induces two kinks in the value function. If a household is 'far away' from the downpayment requirement he just drifts down and is caught in a poverty trap. If another household is just below the requirement, she will reduce consumption and save 'like crazy', and have a high and positive drift, so that she can move to a region where she can afford the house. We thus have multiple stationary distributions, depending on how many households who start in the poverty trap region.

Exercises

On the github page you will find two matlab scripts. One solves the decision problem of the benchmark model using the implicit method and the other solves the decision problem with housing. You are welcome to use this as a guiding tool - but of course you can use any programming language you want (e.g. Julia, Matlab, Fortran (with LAPACK), Python etc).

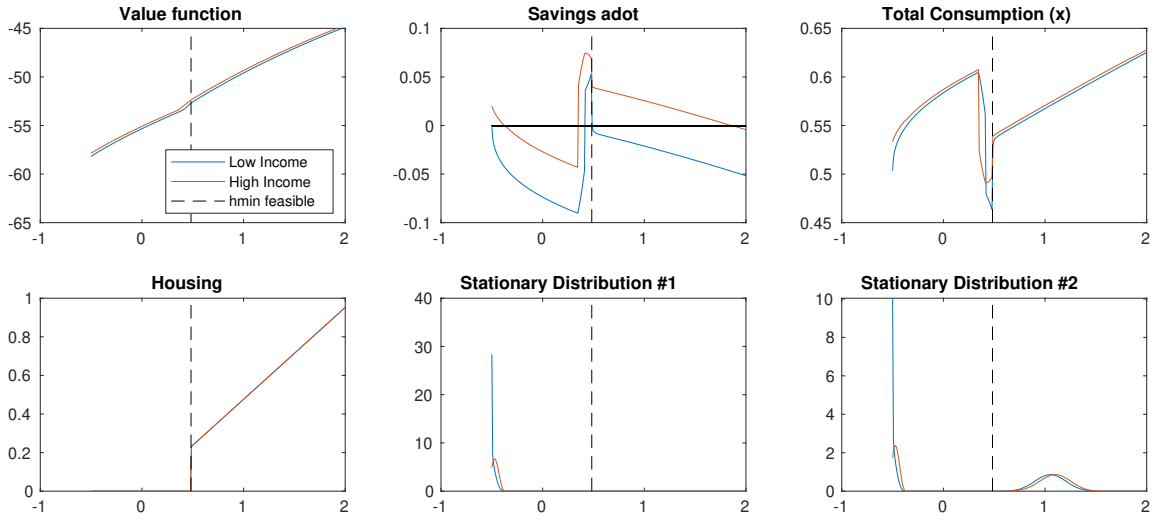
Exercise 1 *Explain why $x_{1,j} = z_{N,j} = 0$, so that $V_{0,1}, V_{N+1,j}$ are never used in equation* (17).

For the rest of the exercises use the parameter values in the posted Matlab script.

Exercise 2 *Rewrite the program that is posted on the webpage to include a switch so that you can use either the implicit or explicit method. Make sure both methods converge*

1. *Time the different methods, which one is faster?*

Figure 2: Solution - Extended Model With Housing



The dashed vertical line represents the first wealth level you can meet the downpayment requirement. Note that both wealth distributions integrate to one, but that in one all the mass is caught in the poverty trap.

2. *Which can you 'break' easier (e.g. making the grid too sparse, increasing γ)?*

3. *What happens when the time-step becomes too large in the explicit method?*

Exercise 3 *Write a script that finds the equilibrium interest rate of the benchmark Ayiagari-model with the parameter values from the benchmark script when the net supply $\bar{b}$ is 0.0 and 0.5. Plot the two distribution and policy functions. Note: Note that net supply = 0 implies that no households can save, since the borrowing constraint $a_{min} = 0$.*

Exercise 4 *Extend the code using the implicit method to have three income levels. Plot the resulting value functions, policy functions and wealth distributions.*

Help the future students and learn Git(hub): *Fork the repo on github, and change the code (e.g. add comments, improve code readability, fix errors) and post a pull request.*

References

Achdou, Y., Han, J., Lasry, J.-M., Lions, P.-L., and Moll, B. (2017a). Income and Wealth Distribution in Macroeconomics: a Continuous-Time Approach.

Achdou, Y., Han, J., Lasry, J.-M., and Moll, B. (2017b). Numerical Methods for Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach.

Barczyk, D. and Kredler, M. (2017). Evaluating Long-Term-Care Policy Options, Taking the Family Seriously. *The Review of Economic Studies*.

Candler, G. (1999). Dynamic Programming. *Finite-Difference Methods for Continuous-Time Dynamic Programming*, pages 55–60.

A Deriving the HJB equation

Let $\beta(\Delta) = e^{-\rho\Delta}$ and the probability of staying in the income state be given by $\pi(\Delta) = e^{-\lambda\Delta}$.

The Bellman equation is:

$$\begin{aligned} V(a_t, y_t) &= \max_c \{u(c)\Delta + \beta(\Delta)\left[\pi(\Delta)V(a_{t+\Delta}, y) + (1-\pi(\Delta))V(a_{t+\Delta}, \tilde{y})\right]\} \\ \text{s.t. } & a_{t+\Delta} = \Delta(y_j + ra_t - c_t) + a_t, \quad a_{t+\Delta} \geq 0 \end{aligned}$$

Next, note that for small Δ we have $\beta(\Delta) \approx (1-\rho\Delta)$ and $\pi(\Delta) \approx 1 - \lambda\Delta$. Substitute into the HJB equation:

$$V(a_t, y_t) = \max_c \{u(c)\Delta + (1-\rho\Delta)\left[(1-\lambda\Delta)V(a_{t+\Delta}, y) + \lambda\Delta V(a_{t+\Delta}, \tilde{y})\right]\}.$$

Subtract $(1-\rho\Delta)V(a, y)$ and rearrange:

$$\begin{aligned} \rho\Delta V(a_t, y_t) = &\max_c \big\{u(c)\Delta \\ &+ (1-\rho\Delta)\left[V(a_{t+\Delta}, y) - V(a_t, y) + \lambda\Delta\left(V(a_{t+\Delta}, \tilde{y}) - V(a_{t+\Delta}, \tilde{y})\right)\right]\big\}. \end{aligned}$$

Divide by Δ, let $\Delta \to 0$ and observe that:

$$\begin{aligned} &\lim_{\Delta\to 0} \frac{V(a_{t+\Delta}, y_t) - V(a_t, y_t)}{\Delta} \\ =&\lim_{\Delta\to 0} \frac{V(\Delta(y_j + ra_t - c_t) + a_t, y_t) - V(a_t, y_t)}{\Delta} \\ =&V_a(a_t, y_t)(y_j + ra_t - y_t) \end{aligned}$$

Which then yields the HJB equation in (4)