Contents

1 Household's Partial Equilibrium Problem **2**

2 The HJB Equation **3**

3 Numerical Solution Without Drift-Splitting **4**

3.1 Upwinding in Unity 4

3.2 Notes on Implementation 5

3.3 Updating Policies for $\dot{b} = 0$ (and Boundary Conditions for b) . 6

3.4 Updating Policies for $\dot{a} = 0$ 8

In what follows we present a new numerical algorithm for solving the partial equilibrium problem of the household with two assets and kinked adjustment costs of the type used in the HANK paper ([Kaplan et al., 2018]). this documents describes the upwind implicit numerical scheme to solve the problem as described here (hereafter KMV) but without splitting the drift of the illiquid asset, and by nonlinear treatment of the boundary conditions. We try to remain close to KMV code in term on notation; please note that the KMV description and note are not exactly those used in the paper [Kaplan et al., 2018]. You can consult [Achdou et al., 2022] for detailed description of the underlying theory and methods.

1 Household's Partial Equilibrium Problem

- The household solves

$$\max_{\{c_t, d_t\}_{t \geq 0}} \mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt$$

- subject to

$$\begin{aligned}
\dot{b}_t &= (1-\xi) w z_t + r^b(b_t) b_t - d_t - \chi(d_t, a_t) - c_t \\
\dot{a}_t &= r^a a_t + \xi w z_t + d_t
\end{aligned}$$

- with constraints

$$a_t \geq 0, \quad b_t \geq \underline{b}$$

- where a_t, b_t denote illiquid and liquid assets, respectively, c_t is consumption, z_t is the idiosyncratic productivity which is considered to be a two-state Poisson process with intensities $\lambda(z, z')$ [1], d_t is the depositing rate and $\chi(.,.)$ the transaction cost function. The wage is denoted by w, the return on illiquid asset is r^a and the return on liquid asset is r^b. Finally we assume that a fraction ξ of income is automatically deposited in the illiquid account (e.g. capturing automatic payroll deductions into a 401(k) account).

- Let's assume the following functional form for the adjustment cost function:

[1]However, the code is written for Poisson processes with any finite number of states N_z

$$\chi(d,a) = \chi_0|d| + \frac{\chi_1}{2}\left(\frac{d}{a}\right)^2 a \tag{1}$$

- The two components of the adjustment cost function have different implications for the household behaviour:
   1. the kinked cost component implies inaction,
   2. the convex component implies finite deposit rates.
- In what follows let $g(d,a)$ denote the net effect of deposit policy d on cash in hand of agent who has illiquid wealth a:

$$g(d,a) = d + \chi(d,a)$$

Conditions on χ parameters:

1. First, $\chi_0, \chi_1 > 0$.
2. Then, assume $r^a < \frac{1-\chi_0}{\chi_1}$ to ensure households won't accumulate illiquid wealth to infinity.
3. Finally, assume $\chi_0 < 1$, otherwise it never makes sense to withdraw. That is, $g(d,a)$ would be non-negative on the whole domain $d \in \mathbb{R}$. (this is also implied by 2 if degenerate cases are ruled out).

2 The HJB Equation

The HJB equation is

$$\begin{aligned}\rho V(a,b,z) = \max_{c,d}\ & u(c) + V_b(a,b,z)((1-\xi)wz + r^b(b)b - d - \chi(d,a) - c) \\ & + V_a(a,b,z)(r^a a + \xi wz + d) \\ & + \sum_{z'} \lambda(z,z')(V(a,b,z') - V(a,b,z))\end{aligned} \tag{2}$$

- The first order conditions are

$$\begin{aligned} u'(c) &= V_b(a,b,z) \\ V_b(a,b,z)(1+\chi_d(d,a)) &= V_a(a,b,z) \end{aligned} \tag{3}$$

- Note that $\chi_d(d,a) =$

$$\begin{cases} \chi_0 + \chi_1 d/a, & d > 0 \\ -\chi_0 + \chi_1 d/a, & d < 0 \end{cases} \tag{4}$$

- Based on the equation above, optimal deposits satisfy

$$d = \left(\frac{V_a}{V_b} - 1 + \chi_0\right)^{-} \frac{a}{\chi_1} + \left(\frac{V_a}{V_b} - 1 - \chi_0\right)^{+} \frac{a}{\chi_1} \tag{5}$$

- Or, equivalently, assuming we know the optimal consumption policy,

$$d = \left(\frac{V_a}{u'(c)} - 1 + \chi_0\right)^{-} \frac{a}{\chi_1} + \left(\frac{V_a}{u'(c)} - 1 - \chi_0\right)^{+} \frac{a}{\chi_1} \tag{6}$$

- In particular, $d = 0$ if $-\chi_0 < \frac{V_a}{V_b} - 1 < \chi_0$ (the inaction region).

3 Numerical Solution Without Drift-Splitting

Before proceeding further, let's introduce notations for three *special* deposit policy $d(a, b, z)$ (and their corresponding $c(a, b, z)$ assuming $\dot{b} = 0$):

- $\mathbf{d_0}, \mathbf{c_0}$: correspond to $d = 0, c = c(d = 0, \dot{b} = 0) = (1 - \xi)wz + r^b(b)b$
- $\underline{\mathbf{d}}, \underline{\mathbf{c}}$: correspond to $\underline{d} = \left(\frac{\chi_0 - 1}{\chi_1}\right) a$, $\underline{c} = c(d = \underline{d}, \dot{b} = 0)$. Note that by assumption 1, $\underline{d} < 0$. This is a point which maximises the cash in hand.
- $\tilde{\mathbf{d}}, \tilde{\mathbf{c}}$: correspond to $\tilde{d} = -(r^a a + \xi wz), \tilde{c} = c\left(d = \tilde{d}, \dot{b} = 0\right)$. This is a policy which makes the a-drift zero (i.e., the threshold at which the sign of the a-drift switches).

3.1 Upwinding in Unity

The general algorithm to upwind without split is very simple; the main idea is to **nest** (rather than **split**) $\dot{a}_t$ inside the upwinding procedure for $\dot{b}_t$. So we consider upwinding of drift of b, but inside each of backward and forward b-drift cases, we take care of upwinding of a.

More specifically, given $V^{n-1}(b_i, a_j, z_k)$ as the current guess (hereafter $V^{n-1}_{i,j,k}$ for more concise notation following KMV)

1. Start with $c^{n,F}$ (that is, use $V_b^{n-1,F}$ to update the consumption policy c^n).
2. Use $c^{n,F}$ with equation (6) *subject to upwinding with respect to the drift of* a to update the deposit policy $d^{n,F}$. More specifically, if we denote by $d(c, V_a) = d(c, V_a; a_j)$ the optimal d given by equation (6),

   (a) If $d(c^{n,F}, V_a^{n,F}) > -(r^a a_j + \xi w z_k)$, then $d^{n,F} = d(c^{n,F}, V_a^{n,F})$.

   (b) If $d(c^{n,F}, V_a^{n,B}) < -(r^a a_j + \xi w z_k)$, then $d^{n,F} = d(c^{n,F}, V_a^{n,B})$.

   (c) If neither of the above holds, then *stay put with respect to a*, that is, $\dot{a} = 0$ which implies $d^{n,F}(b_i, a_j, z_k) = \tilde{d}(a_j, z_k) = -(r^a a_j + \xi w z_k)$.

3. After solving for $d^{n,F}$ using $c^{n,F}$, now plug them back in the drift for b to check whether they are consistent. In particular, if $c^{n,F} + g(d^{n,F}, a_j) < (1-\xi) w z_k + r^b(b_i) b_i$, then we update the optimal consumption and depositing policies as: $c^n = c^{n,F}, d^n = c^{n,F}$.

4. Otherwise, repeat the same procedure for $c^{n,B}$. That is, compute $d^{n,B}$ similar to above, and then if $c^{n,B} + g(d^{n,B}, a_j) > (1-\xi) w z_k + r^b(b_i) b_i$, consider $c^n = c^{n,B}, d^n = d^{n,B}$ as optimal policies, and proceed from there.

5. If none of the two options above led to consistent drift for b, then solve for the optimal policies assuming that we are put with respect to b, i.e., enforce $\dot{b} = 0$. More on that below.

6. Proceed with updating the value function by the conventional upwinding according to drifts computed using these policies.

3.2 Notes on Implementation

How to implement this algorithm, particularly how to make it amenable to vectorisation that is important for code efficiency in languages such as Matlab? For now, we ignore step 5 ($\dot{b} = 0$) and focus on steps 2-4.

- Let's rewrite our algorithm in terms of the notations used in KMV. In particular, d^{FB} denotes the d policy resulting from applying (5) to V_b^F and V_a^B; d^{BB}, d^{BF} and d^{FF} are defined similarly, with the first letter in superscript denoting the direction of finite difference for $V_b = \frac{\partial V}{\partial b}$ and the second letter for the direction of $V_a = \frac{\partial V}{\partial a}$. Moreover, d^B denotes the final upwind policy for d given the backward direction V_b^B is used for estimating the partial derivative of V_b.

- Using that notation, d^F, d^B would be given by

$$d^F = d^{FF} \mathbb{I}_{\left[d^{FF} > \tilde{d}(a,z)\right]} + d^{FB} \mathbb{I}_{\left[d^{FB} < \tilde{d}(a,z)\right]}$$
$$d^B = d^{BF} \mathbb{I}_{\left[d^{BF} > \tilde{d}(a,z)\right]} + d^{BB} \mathbb{I}_{\left[d^{BB} < \tilde{d}(a,z)\right]}$$

- Given d^F, d^B, estimation of optimal depositing policy d would be updated as

$$\begin{aligned} d &= d^B \mathbb{I}_{\left[c^B + g(d^B, a) > (1-\xi)wz + r^b b\right]} + d^F \mathbb{I}_{\left[c^F + g(d^F, a) < (1-\xi)wz + r^b b\right]} \\ &= d^B \mathbb{I}_{\left[\dot{b}(c^B, d^B) < 0\right]} + d^F \mathbb{I}_{\left[\dot{b}(c^F, d^F) > 0\right]} \end{aligned}$$

- Using the notation from KMV, and for comparison, the last equation can be re written as

$$d = d^B \mathbb{I}_{\left[s^{d,B} < -s^{c,B}\right]} + d^F \mathbb{I}_{\left[s^{d,F} > -s^{c,F}\right]}$$

- Obviously, there is no splitting. So it's different from KMV in the sense that, here, the indicators which tell us whether to use d^B or d^F (or none) are the same indicators telling us whether to use c^B or c^F (or none).

- However, as apparent from above, the implementation up to this point is relatively similar. The key difference is that in forming the indicators for d^B, d^F we take $\tilde{d}(a, z)$ as the point of reference, while KMV take 0. Also in forming d, we take the sign of the whole s^b drift as the indicator, while KMV take only the sign of $s^d = -g(d, a)$.

There is still one important piece left unaddressed: policies for the case of $\dot{b} = 0$, which happens at the lower boundary of the b-drift, as well as when none of the two indicators above leads to consistent b-drifts and we want to have a consistent implementation of $\dot{b}_t = 0$.

3.3 Updating Policies for $\dot{b} = 0$ (and Boundary Conditions for b)

In this case the HJB simplifies to

$$\begin{aligned} \rho V(a, b, z) = \max_d \; & u(c(d)) + V_a(a, b, z)(r^a a + \xi wz + d) \\ & + \sum_{z'} \lambda(z, z')(V(a, b, z') - V(a, b, z)) \end{aligned} \tag{7}$$

- where $c(d) = (1-\xi)wz_t + r^b(b)b - g(d, a)$

- FOC for d satisfies

  $$u'(c)g'(d) = V_a(a, b, z) \implies u'\Big((1-\xi)wz + r^b(b)b - g(d, a)\Big)(1 + \chi_d(d, a)) = V_a(a, b, z)$$

  with $\chi_d(d, a)$ given by (4).

- Note that looking at the interval $[\underline{d}, \infty)$ there should exist a unique solution "for updating optimal d". Assuming that $V_a(a, b, z)$ is a positive number, then the LHS is zero at $\underline{d}$, and it's strictly increasing in d. So there should be a unique solution for it "unless" the solution is "lost" in the jump that happens at 0, in that case we update our approximation of the optimal d to be zero (case 2 below). In other words, (LHS - RHS) of the FOC above is increasing but not continuous.

- The LHS is not only non-linear but kinked, which makes it a bit messy to solve. To solve for d non-linearly using this equation, two things should be taken care of; first whether to use V_a^F or V_a^B to properly upwind, second the jump in $(1+\chi_d)$ (artefact of the kink in adjustment comes).

Now to update the optimal deposit policy for the case of $\dot{b} = 0$, one can proceed with these cases:

1. Either $u'(c_0)(1+\chi_0) < V_a^F$: In this case solve for optimal d on $[0, d^{max})$.
2. Or $\frac{V_a^F}{1+\chi_0} < u'(c_0) < \frac{V_a^F}{1-\chi_0}$, in which case optimal d $= 0$.
3. Finally, if $u'(c_0)(1-\chi_0) > V_a^F$
   (a) If $\underline{d} > \tilde{d}$, solve the FOC for d, again using V_a^F, on $[\underline{d}, 0]$.
   (b) Else
      i. If $u'(\tilde{c})(1 + \chi_d(\tilde{d}, a) < V_a^F$, then: solve for d on $[\underline{d}, \tilde{d}]$ using V_a^B.
      ii. If $u'(\tilde{c})(1 + \chi_d(\tilde{d}, a) < V_a^F$, solve for d on $[\tilde{d}, 0]$ using V_a^F.
      iii. Otherwise, optimal $d = \tilde{d}$ (that is, a-drift is zero).

Some comments on implementation:

- If this seems a bit obscure, we encourage you to draw a graph of $g(d, a)$ and designate the points $\tilde{d}, \underline{d}$, and d_0 on it, to see why the steps above make sense.

- In each case above, after solving for the optimal d, the consumption policy c can be computed using the $\dot{b}_t = 0$ condition, i.e.,

$$c(b, a, z) = (1 - \xi)wz + r_b(b)b - g(a, d)$$

- Variables in the code: in the accompanied Matlab code, $\tilde{d}$ is denoted by $d_{zerodrift}$ and c_0, d_0 denote the c and d policies for the case in which $\dot{b}_t = 0$, (rather than $d = 0$).
- To solve for d (non-linearly), in cases 1, 3a and 3b above, one case use one of the bracketing methods, e.g. Brent's method. The intervals suggested should give a negative evaluation of the (LHS - RHS) of the FOC on the lower bound, and positive on the upper bound, assuming V_a is positive. In the Matlab code we use `fzero`.
- To code can be easily vectorised, either using Matlab built-in vectorisation functions such as `funarray`, or by implementing a user defined Brent's method. We have not used that for better readability.

Again, this can be vectorised similar to above.

3.4 Updating Policies for $\dot{a} = 0$

Which we use at a-boundaries or where neither V_a^F nor V_a^B leads to consistent policies. This bit is straightforward. Set $d = -(r^a a + \xi wz)$, then solve for c by upwinding with respect to $\dot{b}$.

Note: As long as V is concave and monotone in each of assets separately, the above scheme is unambiguous, monotone and upwind. See [Achdou et al., 2022] for proofs.

References

[Achdou et al., 2022] Achdou, Y., Han, J., Lasry, J.-M., Lions, P.-L., and Moll, B. (2022). Income and wealth distribution in macroeconomics: A continuous-time approach. *The review of economic studies*, 89(1):45–86.

[Kaplan et al., 2018] Kaplan, G., Moll, B., and Violante, G. L. (2018). Monetary policy according to hank. *American Economic Review*, 108(3):697–743.