# The Nested–Drift Algorithm

## A New Solution Approach to Continuous Time Problems with Multiple Endogenous State Variables

Soroush Sabet  Patrick Schneider

September 5, 2025

*Preliminary and incomplete*

**Abstract**

We introduce a new algorithm for solving continuous time problems with multiple endogenous state variables in a finite–difference scheme. The algorithm extends the logic of up–winding to its logical extreme in a way that meets the necessary conditions for local convergence (Barles and Souganidis, 1991). It is consistent because the directions of state–drift are never in conflict, and constraints are applied non–linearly. It is efficient because it nests the search for these state–drifts in a way that ensures costly root–finding is only performed after other alternatives are exhausted. The algorithm improves on the existing 'split–drift' approach from (Kaplan et al., 2018; Achdou et al., 2022), which can fail under some parameter settings because it is not guaranteed to be consistent.

# 1 Introduction

We present a new numerical algorithm for solving finite–difference, continuous time problems of agents with two endogenous state variables. The key to the algorithm is to nest the treatment of the state drift within each other. Doing so ensures there are no conflicting assumptions in drift, and allows us to treat constraints non–linearly. As a result, the algorithm is consistent and monotone, and so local convergence to a viscosity solution is guaranteed, following Barles and Souganidis (1991).

The most common example of the model to which this algorithm applies is a household with two assets and adjustment costs for transacting between them, as in Kaplan et al. (2018). In the following, we use a simplified version of the household block in their model to demonstrate the algorithm.

The algorithm works by extending the approach of upwinding (Achdou et al., 2022) to its logical extreme—at each point in the state–space, optimal policy functions are calculated using numerical derivatives of the value function that correspond to the endogenous direction in drift for *both* states, ex–post[1].

Dealing with drift directions state–by–state also allows us to avoid conflicting assumptions, and also to impose constraints in a more non–linear way. The combination of all these things is a scheme that is consistent and monotone, and so converges for a starting guess sufficiently close to the solution.

To get a good starting guess, we employ a 'tentative–guess' algorithm, which attempts a solution, but slows updating dramatically for a period when one cannot be reached, after which it accelerates exponentially again; repeating this process recursively, restarting after failures at the last slow update, tends to lead to convergence. Using such procedures seem to be folk–wisdom in the profession, though we are not aware of a proof for why they work, and nor do we provide one.

The paper proceeds as follows. In Section 3 we detail the model that will be our working example. In Section 4 we detail the solution algorithm, providing a proof that the nested–drift algorithm is consistent and monotone. We implement the algorithm in Section 5 and plot the results. In Section 6 we compare our algorithm to the alternative 'split–drift' approach in Kaplan

[1]This is not the case in the drift–splitting algorithm proposed by Kaplan et al. (2018), which we find leads to solution failures for some parameters that our algorithm can solve.

et al. (2018), showing that the our approach is robust to parameter choices under which the latter fails. Section 7 concludes.

# 2 General approach

For a problem with vectors x controls, y endogenous states, and z exogenous states. Update solution is value derivatives, policies and drift. Interested in problems where policies are analytical functions of value derivatives. If you know one, you know the other two. With analytical policy solutions, we can 'know' value derivatives from the VFI process and then find the others to update. Exhausting the two numerical derivative options means we know drift, and can work backward to find policies, conditional on what these imply for value derivatives. Identification requires using all this info as a system. Guessing the value derivative as the midway point and also imposing zero drift imposes too much structure - the policy foc won't hold except by coincidence. This is not the case with one state and one control, but it is with more (probably why we failed yesterday). Cases: add a control (labour supply), add a state (housing) etc. Is there a case with three where one of the controls affects all the states?

## 2.1 Upwinding with one endogenous state

Description of upwinding in general. The solution to the problem is a system of equations defining the value, policies, and drift. For example, with just one endogenous state (and an exogenous one thrown in for good measure), the solution is as follows wheer $f(\cdot)$ is the state transition equation, $g(\cdot)$ arises from the policy equation's FOC, and $\mathcal{A}$ is the infitessimal generator that captures exogenous state transitions:

$$\rho v(a,z) - \partial_t v(a,z) = u(c(a,z)) + \partial_a v(a,z) \cdot \dot{a}(a,z) + \mathcal{A}\left[v\right](a,z) \tag{1}$$

$$\dot{a}(a,z) = f(a,z,c(a,z)) \tag{2}$$

$$c(a,z) = g(\partial_a v(a,z)) \tag{3}$$

In the numerical solution in the updating process we do not know $v(a, z)$, nor its derivatives. We do know the discretised guess for the value $v^{n+1}(a_i, z_j)$ and so we can estimate the derivatives. The system of equations describing the semi–implicit numerical up–winding solution for iteration $n$ at the point $(a_i, z_j)$ in the state–space is therefore the existing equations as well as the new conditions defining the partial derivative

$$\partial_a v^n(a_i, z_j) \in \begin{cases} [V_a^F] & \text{if } \dot{a}(a_i, z_j) > 0 \\ [V_a^B] & \text{if } \dot{a}(a_i, z_j) < 0 \\ [V_a^F, V_a^B] & \text{if } \dot{a}(a_i, z_j) = 0 \end{cases} \tag{4}$$

$$V_a^F = \frac{v^{n+1}(a_{i+1}, z_j) - v^{n+1}(a_i, z_j)}{a_{i+1} - a_i} \tag{5}$$

$$V_a^B = \frac{v^{n+1}(a_i, z_j) - v^{n+1}(a_{i-1}, z_j)}{a_i - a_{i-1}} \tag{6}$$

Six equations in six unknowns. If only one of the cases in Equation 4 can hold, then we can rule out the expensive ones first. In this case, the expensive option is where the the third case, where the partial derivative is set–defined. With only one endogenous state variable, it's not *so* expensive because in this case we know consumption from the endogenous state transition equation, and can back out the partial derivative from the FOC.

## 2.2 With multiple endogenous state variables

When multiple policies show up in drifts that's the difficult bit. Put those drifts at the top of the hierarchy. Put the drifts with only one policy influence at the bottom - nothing expensive in these ones, either base on drift directions or the zero–drift condition. Root–finding is only necessary when there are multiple policies in a drift equation. As long as policy functions are one to one and onto then they're invertible to find derivatives in root–finding.

Figure a general way to frame the problem. General conditions for when this works. What if you increase controls, but keep endogenous states fixed? What about the opposite?

What if you combine with discrete choices? Intense jump like with HJBVI?

# 3 The model

In this section we detail the model we will use for our worked example of the algorithm. The logic of the algorithm extends to any setting with two endogenous state variables.

A household chooses consumption and transfers between their liquid and illiquid assets to maximise the present–value of utility.

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

subject to budget constraints

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

with borrowing limits

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

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 $\lambda(z, z')$ [2], $d_t$ is the depositing rate and $\chi(d, a)$ 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 $\xi$ of income is automatically deposited in the illiquid account (e.g. capturing automatic payroll deductions into a retirement account).

Following Kaplan et al. (2018), we assume transactions between the two accounts attract adjustment costs with the functional form:

$$\chi(d, a) = \chi_0 |d| + \frac{\chi_1}{2} \left(\frac{d}{a}\right)^2 a \tag{7}$$

Where $\chi_0 \in (0, 1)$ and $\chi_1 > 0$. The two components of the adjustment cost function have different implications for the household behaviour:

1. The linear cost component creates inaction,

[2] The code is written for Poisson processes with any finite number of states $N_z$

2. The convex component creates finite deposit rates, eliminating jumps

In what follows let $g(d,a)$ denote the net effect of deposit policy $d$ on the liquid resources of an agent who has illiquid wealth $a$

$$g(d,a) = d + \chi(d,a) \tag{8}$$

Note that there is a natural minimum to the optimal deposit policy

$$\underline{d} = \frac{\chi_0 - 1}{\chi_1} a < 0 \tag{9}$$

This is the point beyond which the marginal cost of withdrawing is greater than 1; any transfers from the illiquid to the liquid asset lower than this point will actually reduce liquid assets, destroying resources.

Further, as identified in Kaplan et al. (2018), it is necessary to set $r^a < \frac{1-\chi_0}{\chi_1}$ to ensure households don't optimally accumulate illiquid wealth to infinity. If this condition is not met, the flow return on the illiquid asset ($r^a a$) is greater than $\underline{d}$[3].

## 3.1 The HJB Equation & analytical solution

The HJB equation for the problem is

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

The first order conditions are

$$u'(c) = V_b(a,b,z) \tag{10}$$

$$V_b(a,b,z)(1 + \chi_d(d,a)) = V_a(a,b,z) \tag{11}$$

where

$$\chi_d(d,a) = \begin{cases} \chi_0 + \chi_1 d/a, & d > 0 \\ -\chi_0 + \chi_1 d/a, & d < 0 \end{cases}$$

[3]There are inflows from the wage as well, but these are assumed minor at higher asset levels, and so ignored.

And rearranging these we arrive at the updating equations for consumption:

$$c = (u')^{-1}(V_b) \tag{12}$$

and deposits

$$d = \left(\frac{V_a}{V_b} - 1 + \chi_0\right)^{-} \frac{a}{\chi_1} + \left(\frac{V_a}{V_b} - 1 - \chi_0\right)^{+} \frac{a}{\chi_1} \tag{13}$$

Where $(\cdot)^+ := \max\{\cdot, 0\}$ and $(\cdot)^- := \min\{\cdot, 0\}$. Note that this policy implies the inaction region $d = 0$ if $-\chi_0 < \frac{V_a}{V_b} - 1 < \chi_0$. Combining the two FOC, we can frame the deposit policy in terms of optimal consumption

$$d = \left(\frac{V_a}{u'(c)} - 1 + \chi_0\right)^{-} \frac{a}{\chi_1} + \left(\frac{V_a}{u'(c)} - 1 - \chi_0\right)^{+} \frac{a}{\chi_1} \tag{14}$$

These conditions describe interior solutions. The state boundary constraints are dealt with in the algorithm, rather than imposed here, our first departure from the standard approach (Achdou et al., 2022).

# 4 Numerical Solution

We implement this model in a discrete approximation with $(I, J, N_z)$ denoting the number of liquid grid points, illiquid grid points, and exogenous income states, respectively[4]. The algorithm uses the standard structure detailed in Achdou et al. (2022) where we (1) start with a guess for the value, (2) update policies using the numerical derivative calculated from the guessed value, then (3) use these policies to create a discrete upwind transition matrix[5], and (4) use this matrix to solve implicitly for the updated value function, repeating this process until convergence in the value.

The insight we bring is to Step (2). Solution algorithms need to ensure that numerical derivatives used to calculate policies are appropriate, but there are infinitely many derivatives one could use between the bounds of a forward and backward derivative. To this end, up–winding is recommended—using

[4]These grids can be linear or non-linear, an option allowed in the provided code

[5]More precisely, discretised infinitesimal generator for the value function.

the numerical derivative in the direction that leads to a policy function that reproduces endogenous state drift in the originally assumed direction. In a one–asset environment, an example of an upwind scheme is to assume first that the saving rate will be positive, and so to use the forward numerical derivative of the value; if the consumption policy based on this derivative delivers positive savings, then this is the policy that is used, otherwise the next step is to assume negative saving, and so on. Up–winding is simple with one endogenous state variable but it becomes complicated with more than one.

The key insight in our approach is to extend the logic of up–winding in a comprehensive way for more endogenous state variables, and to do this efficiently by delaying expensive root–finding computations until we are sure they're needed. The following details the step–by–step implementation of the algorithm for the model outlined in Section 3.

## 4.1 Updating the policy functions: the nested–drift algorithm

For a starting value function $V_{ijk}^{n-1} := V^{n-1}(b_i, a_j, z_k)$, and for each point[6] in the grid $(ijk)$:

1. **Calculate key objects**:
   - Zero illiquid drift deposit $\tilde{d}_{ijk} = -(r^a a_j + \xi w z_k)$
   - Minimum rational deposit $\underline{d}_{ijk} = \frac{\chi_0 - 1}{\chi_1} a_j$
   - Directional derivatives

$$V_b^F = \frac{V_{(i+1)jk}^{n-1} - V_{ijk}^{n-1}}{b_{i+1} - b_i} \ \forall i < I \text{ and } V_b^B = \frac{V_{ijk}^{n-1} - V_{(i-1)jk}^{n-1}}{b_i - b_{i-1}} \ \forall i > 1$$

$$V_a^F = \frac{V_{i(j+1)k}^{n-1} - V_{ijk}^{n-1}}{a_{j+1} - a_j} \ \forall j < J \text{ and } V_a^B = \frac{V_{ijk}^{n-1} - V_{i(j-1)k}^{n-1}}{a_j - a_{j-1}} \ \forall i > 1$$

2. **Assume forward liquid drift** (skip if $i = I$ due to state constraint):

[6] In practice, this can be vectorised, and the provided Matlab code does so as much as is feasible, or parallelised.

First assume forward liquid drift, where the policies are $(c^F, d^F)$ and they should lead to

$$\dot{b}_{ijk} = r^b_{ijk} b_i + (1-\xi) w z_k - c^F - d^F - g(d^F, a_j) > 0$$

Find the policies (suppressing subscripts for the remainder of the algorithm for clarity, as they all read $ijk$):

(a) Consumption comes from FOC equation (12)

$$c^F = (u')^{-1}\left(V_b^F\right)$$

(b) There are two possible deposit policies, both from FOC equation (13), depending on the illiquid drift they create. We label these policies $d^{FF}$ and $d^{FB}$ where the first superscript denotes the drift of the liquid asset they're associated with and the second the drift in the illiquid asset. Because we have assumed liquid drift is forward, the two potential deposit policies are therefore

$$d^{FF} = \left(\frac{V_a^F}{V_b^F} - 1 + \chi_0\right)^- \frac{a}{\chi_1} + \left(\frac{V_a^F}{V_b^F} - 1 - \chi_0\right)^+ \frac{a}{\chi_1}$$
$$d^{FB} = \left(\frac{V_a^B}{V_b^F} - 1 + \chi_0\right)^- \frac{a}{\chi_1} + \left(\frac{V_a^B}{V_b^F} - 1 - \chi_0\right)^+ \frac{a}{\chi_1}$$

Impose illiquid state constraints if $j \in \{1, J\}$

$$d^{FF}_{j=J} = \tilde{d}$$
$$d^{FB}_{j=1} = \tilde{d}$$

Now define $d^F$ using the policy consistent with its assumed drift

$$d^F = \begin{cases} d^{FF} & \text{if } d^{FF} > \tilde{d} \\ d^{FB} & \text{if } d^{FB} < \tilde{d} \\ \tilde{d} & \text{otherwise} \end{cases}$$

These conditions are mutually exclusive and collectively exhaustive, provided $V$ is concave in $a$: for a given $V_b$, equation (13) is increasing in $V_a$; with $V$ concave in $a$ then $V_a^B > V_a^F$ so $d^{FF} < d^{FB}$. Hence the possible cases are $\tilde{d} < d^{FF} < d^{FB}$, $d^{FF} < d^{FB} < \tilde{d}$, or neither ($d^{FF} < \tilde{d} < d^{FB}$), as above.

Now that the conditional policies are known, check whether they're consistent with the assumption of forward liquid drift.

(a) Define the asset drift coming from both policies

$$s_b^F = r^b b + (1-\xi)wz - c^F - g(d^F, a)$$
$$s_a^F = r^a a + \xi wz + d^F$$

(b) If $s_b^F > 0$ then the assumption is upheld, and these are the true policies $c_{ijk} = c_{ijk}^F$, $d_{ijk} = d_{ijk}^F$, $s_{a,ijk} = s_{a,ijk}^F$ and $s_{b,ijk} = s_{b,ijk}^F$ Move on to the next point in the state space.

(c) Otherwise if $s_b^F \leq 0$ then the assumption is not upheld, and we move on to check for backward liquid drift.

3. **Assume backward liquid drift** (skip if $i = 1$ due to state constraint):

   Apply equivalent steps to the above, but using the backward liquid derivative in place of forward i.e. replacing $V_b^F$ with $V_b^B$.

   If the conditional policies are consistent with backward drift, these are the correct policies. Otherwise, move on to the zero liquid drift assumption.

4. **Assume zero liquid drift**:

   Assuming zero liquid drift, consumption is determined by the deposit policy

$$c^0 = r^b b + (1-\xi)wz - g(d^0, a)$$

   so this deposit policy ($d^0$) is all we need to find. Furthermore, because we know the consumption policy, we can infer the value derivative with respect to the liquid asset from equation (10)[7]

$$V_b = u'\left(c^0\right)$$

   The optimal deposit policy $d^0$ will therefore be defined implicitly by substituting these objects into the no–arbitrage first order condition, equation (11)

$$u'\left(r^b b + (1-\xi)wz - g\left(d^0, a\right)\right) g_d\left(d^0, a\right) = V_a$$

[7]This will respect the FOC because $c^0 \in [c^B, c^F]$, proof Appendix A.

To help identify this implicit policy, define a state–dependent function of the deposit policy

$$F\left(d, V_a\right) = u'\left(r^b b + (1-\xi)wz - g\left(d, a\right)\right) g_d\left(d, a\right) - V_a$$

This function will be

- negative at $d = \underline{d}$ because $g_d(\underline{d}, a) = 0$ and $V_a > 0$,
- infinitely positive for finite $d^* > 0$ due to the Inada condition, and
- monotonically increasing in the range $d \in [\underline{d}, d^*]$

So a solution $F(d^0, V_a) = 0$ is guaranteed, but we need to go carefully about finding it because of kinks caused by the cost adjustment cost function at $d = 0$, discrete approximation errors around $d = \tilde{d}$, and the need to use the appropriate directional derivative in place of $V_a$.

We proceed by checking different segments of the potential range of $d$ one at a time, to cover all the possible special cases created by the adjustment inaction region and the difference in forward and backward illiquid value derivatives. The cases, and the order they're considered in, are outlined in table (1).

Table 1: Cases when $\dot{b} = 0$

| | $\dot{a} > 0$ | $\dot{a} = 0$ | $\dot{a} < 0$ |
|---|---|---|---|
| $d > 0$ | (1) | n.a. | n.a. |
| $d = 0$ | (2) | n.a. | n.a. |
| $d < 0$ | (3) | (4) | (5) |

The algorithm proceeds as follows, for all $a$ unless at the top of the illiquid state grid.

(a) Check whether $F(0, V_a^F) < 0$. If so, $d^0$ should be positive; use a root-finding method to find $F(d^0, V_a^F) = 0$ in the range $d \in [0, \bar{d}]$ where $\bar{d}$ is the deposit rate that would imply negative consumption if $\dot{b} = 0$. Otherwise, continue.

(b) Check whether $F(-\varepsilon, V_a^F) < 0$ for a vanishingly small $\varepsilon$. If so (combined with the previous results) $d^0 = 0$. Otherwise, continue.

(c) If $\tilde{d} > \underline{d}$

i. Check whether $F(\tilde{d}, V_a^F) < 0$. If so, use a line–search method to find $F(d^0, V_a^F) = 0$ in the range $d \in [\tilde{d}, 0]$. Otherwise, continue.

ii. Check whether $F(\tilde{d}, V_a^B) < 0$. If so, $d^0 = \tilde{d}$. Otherwise, continue.

iii. If $a = 0$ then impose the state constraint $d^0 = \tilde{d}$. Otherwise, continue.

iv. Use a line–search method to find $F(d^0, V_a^B) = 0$ in the range $d \in [\underline{d}, \tilde{d}]$

(d) If $\tilde{d} \leq \underline{d}$

i. Check whether $F(\underline{d}, V_a^F) < 0$. If so, use a line–search method to find $F(d^0, V_a^F) = 0$ in the range $d \in [\underline{d}, 0]$. Otherwise, continue.

ii. Use a line–search method to find $F(d^0, V_a^F) = 0$ in the range $d \in [\underline{d}, \tilde{d}]$

If $a$ is at the top of the illiquid state grid, we need to impose state constraint where appropriate.

(a) Check whether the solution is in the down–drift region i.e. both $\tilde{d} > \underline{d}$ and $F(\tilde{d}, V_a^B) > 0$ are satisfied. If so, use a line–search method to find $F(d^0, V_a^B) = 0$ in the range $d \in [\underline{d}, \tilde{d}]$. Otherwise, continue.

(b) Set $d^0 = \tilde{d}$ to ensure no upward drift at the top end of the state grid (note this will sometimes mean setting $d^0 < \underline{d}$).

With $d^0$ defined, we can calculate $c^0$, and verify that

$$V_b^F \leq u'\left(c^0\right) \leq V_b^B$$

This is important because it ensures the FOC in equation 12 still hold. A proof of why this is true is provided in the appendix.

Finally, commit the results to the policy functions $c = c^0, d = d^0$ and the drift variables $s_b, s_a$ using their formulae.

After this process is complete, we have policies $(c, d, s_b, s_a)$ defined at each point in the state space.

## 4.2 Finding the drift matrix

Once the policies are known, construct the 'drift matrix' $A$ that describes the discrete, stacked HJB equation:

$$\mathbf{v} = u(\mathbf{c}) + A\mathbf{v}$$

where $A$ is made up of three types of drift: $BB$ describes the drift in the liquid asset, $AA$ the drift in the illiquid one, and $\Lambda$ the Poisson transitions between idiosyncratic states.

$$A = BB + AA + \Lambda$$

and where we build each separately. The provided code does this efficiently, by exploiting the structure of the arrays and sparse matrix objects, but the logic for each is outlined below.

**Liquid drift** $BB$ Each row of $BB$ corresponds to a point in the state–space indexed by $ijk$, and relates to a point in the liquid drift policy $s_{b,ijk}$. All elements of this row are zero except, potentially, the diagonal and its two adjacent cells; these cells have the values:

$$\begin{bmatrix} \dots & \frac{s_b \mathbb{1}_{s_b<0}}{b_i - b_{i-1}} & -\left(\frac{s_b \mathbb{1}_{s_b<0}}{b_i - b_{i-1}} + \frac{s_b \mathbb{1}_{s_b>0}}{b_{i+1} - b_i}\right) & \frac{s_b \mathbb{1}_{s_b>0}}{b_{i+1} - b_i} & \dots \end{bmatrix}$$

At the two ends of the liquid grid, i.e. where $i \in \{1, I\}$, only one of these drift directions is possible. That is, where $i = 1$ the diagonal and the cell to its right have the values

$$\begin{bmatrix} \dots & -\frac{s_b \mathbb{1}_{s_b>0}}{b_{i+1} - b_i} & \frac{s_b \mathbb{1}_{s_b>0}}{b_{i+1} - b_i} & \dots \end{bmatrix}$$

And where $i = I$ the diagonal and the cell to its left have the values

$$\begin{bmatrix} \dots & \frac{s_b \mathbb{1}_{s_b<0}}{b_i - b_{i-1}} & -\frac{s_b \mathbb{1}_{s_b<0}}{b_i - b_{i-1}} & \dots \end{bmatrix}$$

**Illiquid drift** $AA$ Each row of $AA$ corresponds to a point in the state–space indexed by $ijk$, and relates to a point in the illiquid drift policy $s_{a,ijk}$. All elements of this row are zero except, potentially, the diagonal and the cells $I$ positions to the left and right of the diagonal; these cells have the values:

$$\begin{bmatrix} ... & \frac{s_a \mathbb{1}_{s_a<0}}{a_j - a_{j-1}} & ... & -\left(\frac{s_a \mathbb{1}_{s_a<0}}{a_j - a_{j-1}} + \frac{s_a \mathbb{1}_{s_a>0}}{a_{j+1} - a_j}\right) & ... & \frac{s_a \mathbb{1}_{s_a>0}}{a_{j+1} - a_j} & ... \end{bmatrix}$$

At the two ends of the illiquid grid, i.e. where $j \in \{i, J\}$, only one of these drift directions is possible. That is, where $j = 1$ the diagonal and the cell $I$ positions to its right have the values

$$\begin{bmatrix} ... & -\frac{s_a \mathbb{1}_{s_a>0}}{a_{j+1} - a_j} & ... & \frac{s_a \mathbb{1}_{s_a>0}}{a_{j+1} - a_j} & ... \end{bmatrix}$$

And where $j = J$ the diagonal and the cell $I$ positions to its left have the values

$$\begin{bmatrix} ... & \frac{s_a \mathbb{1}_{s_a<0}}{a_j - a_{j-1}} & ... & -\frac{s_a \mathbb{1}_{s_a<0}}{a_j - a_{j-1}} & ... \end{bmatrix}$$

**Idiosyncratic state transitions** $\Lambda$ The idiosyncratic state transition matrix is built the usual way, for example assuming $N_z = 2$ and the Poisson transition rates out of each state are $(\lambda_1, \lambda_2)$

$$\Lambda = \begin{bmatrix} -\lambda_1 & \lambda_1 \\ \lambda_2 & -\lambda_2 \end{bmatrix} \otimes \mathcal{I}$$

Where $\mathcal{I}$ is a $I \times J$ identity matrix.

## 4.3 Updating the value

With the drift matrix in hand, update the value function using the implicit method, for some choice of large $\Delta$

$$\mathbf{v}^n = \left[I(\rho + 1/\Delta) - A\right]^{-1} \left[u(\mathbf{c}) + \mathbf{v}^{n-1}/\Delta\right] \tag{15}$$

Finally, check convergence in $\mathbf{v}$ by some criterion (we use the maximum absolute difference) and iterate if not yet converged.

### 4.4 Finding a good starting guess

The convergence properties of an implicit scheme like this are only local to the solution (Barles and Souganidis, 1991). Hence, starting with a good guess is key to finding a solution, but this can be difficult with an unfamiliar problem. To help with this, we have found the following routine helps improve a guess from a bad starting point, to one closer to the truth.

1. Start with a guess that, at the least, has curvature in both dimensions of the endogenous state variables e.g. $V^0 = u\left(r^b b + r^a a + (1-\xi)wz\right)/\rho$
2. Try to solve the problem with a very high update speed e.g. $\Delta = e^{10}$. If this converges to a solution, then there is no issue.
3. If an update fails along the way (i.e. it yields a value with $V_b < 0$, complex solutions, or some other issue), then reduce the update speed substantially to some base level, for example to $\Delta = \underline{\Delta} = 0.1$, and start the next round from $V^0$.
4. Keep updating for a while (e.g. 20 iterations) at this lower $\underline{\Delta}$, and then store the point you get to as a new $V^0$, reset $n = 1$, and resume updating, scaling up the $\Delta$ again (e.g. $\Delta = \underline{\Delta} \times \exp(n)$). Keep going until either convergence to a solution, or failure, in which case return to step 3.

We don't have a proof for why this works, but it works very well in practice[8].

## 5 Implementation

Matlab codes implementing the algorithm are provided online here[9]. We use the following economic parameters

$$\sigma = 2,\ \rho = 0.05,\ (\chi_0, \chi_1) = (0.03, 2), \xi = 0.1$$

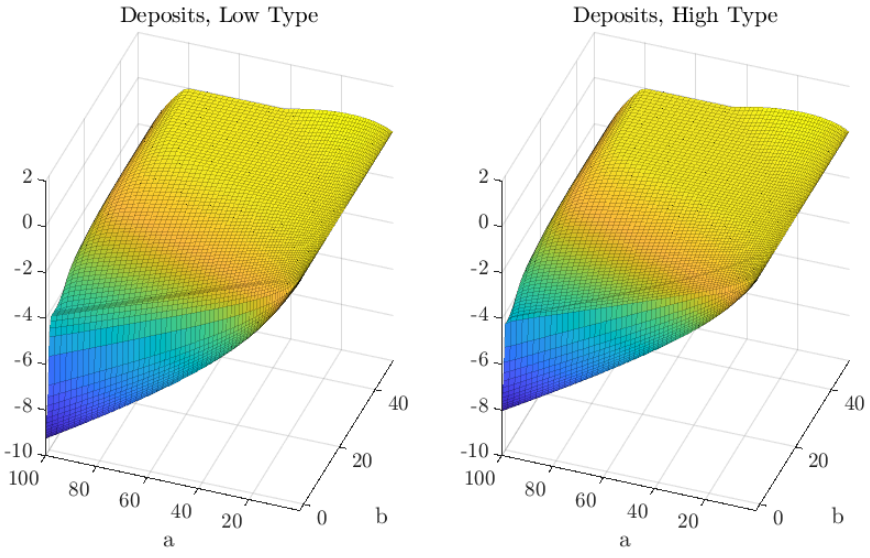
[8] With an explicit update scheme, setting $\Delta = 1$ is equivalent to VFI in discrete time if the Courant–Friedrichs–Lewy condition holds (Rendahl, 2022), and so the guaranteed convergence properties in that environment (Stokey, 1989) apply. This is not the case with an implicit scheme, in which $\Delta$ has no particular significance at any value, yet seems to work in practice.

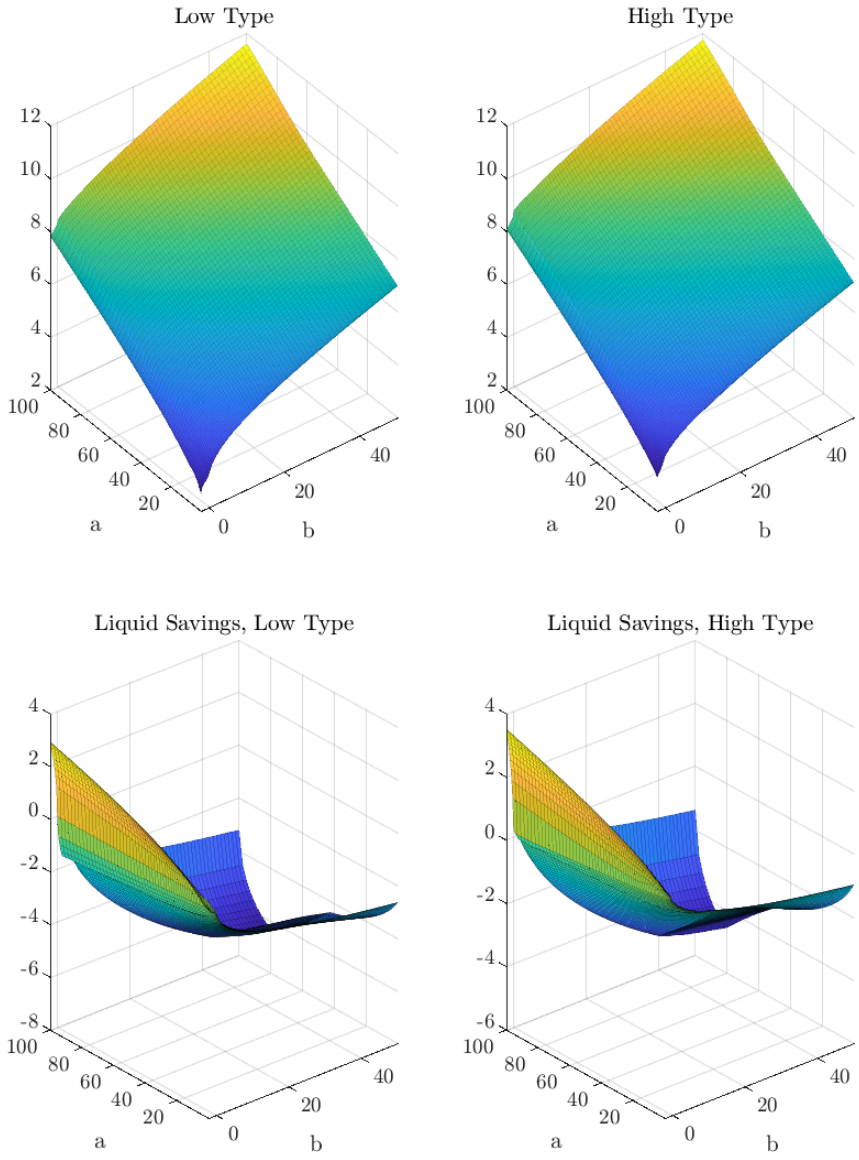
[9] As noted above, these codes owe a debt to KMV, borrowing some lines and notation.

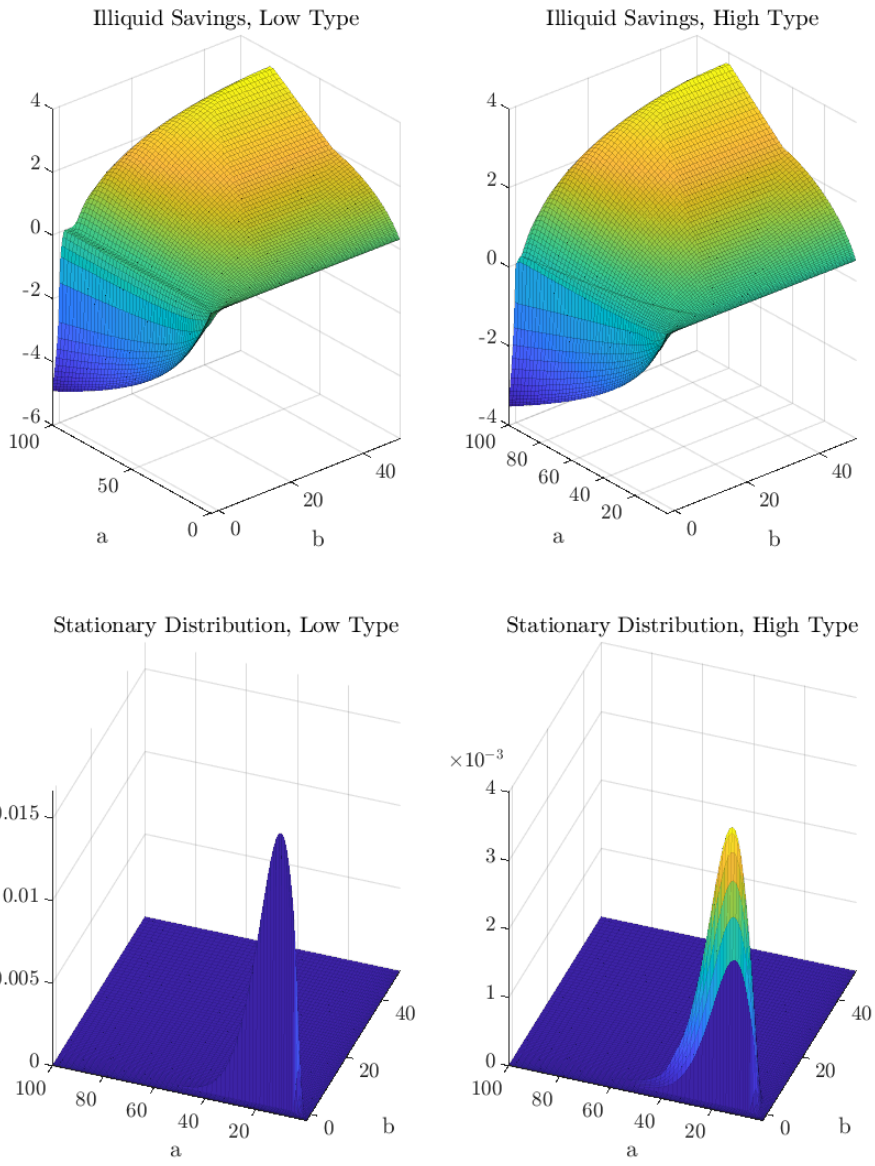
and prices

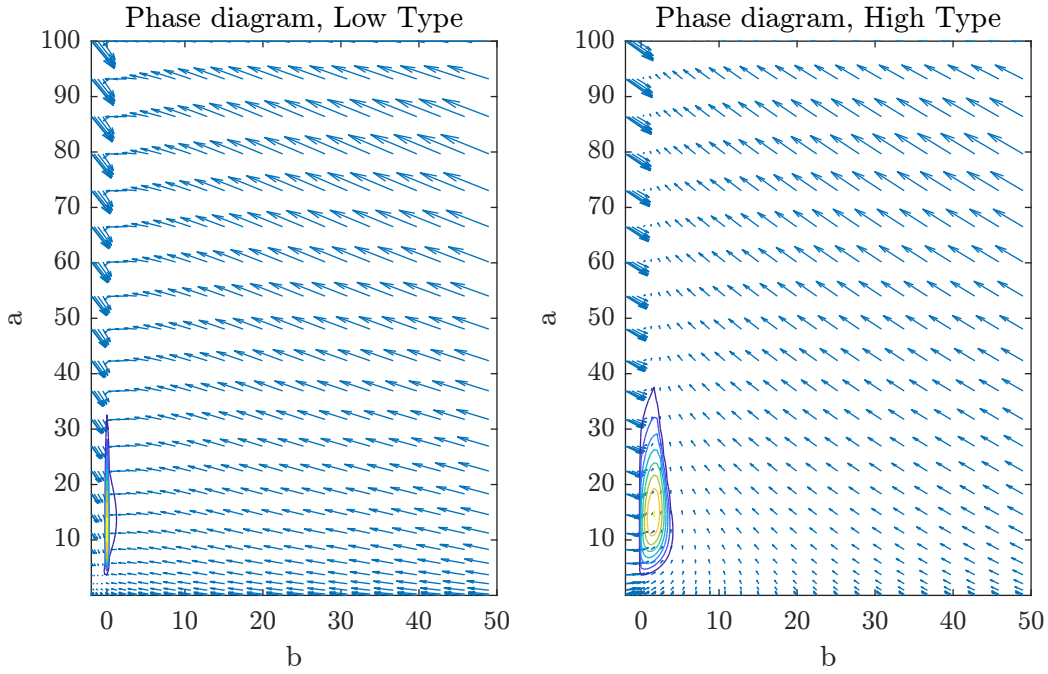
$$r^b = \begin{cases} 0.03 & b \geq 0 \\ 0.12 & b < 0 \end{cases}, \; r^a = 0.04, \; w = 4$$

For the idiosyncratic states we use $z \in \{0.8, 1.3\}$ with $\lambda_{z',z} = 1/3$. For the endogenous states, we use a linear grid of $I = 80$ points for the liquid asset on the range $b \in [-2, 50]$ and a nonlinear grid of $J = 70$ points over the interval $a \in [0, 100]$ for the illiquid asset. Stationary solutions are plotted below.




Low Type
12
10
8
6
4
2
100
80
60
40
20
a
0
20
40
b
High Type
12
10
8
6
4
2
100
80
60
40
20
a
0
20
40
b
Liquid Savings, Low Type
4
2
0
-2
-4
-6
-8
100
80
60
40
20
a
0
20
40
b
Liquid Savings, High Type
4
2
0
-2
-4
-6
100
80
60
40
20
a
0
20
40
b


Illiquid Savings, Low Type
Illiquid Savings, High Type
Stationary Distribution, Low Type
Stationary Distribution, High Type
a
b
×10^−3




# 6 Accuracy and Stability

*Coming soon.*

# 7 Conclusion

*Coming soon.*

# A Proof that $c^0 \in [c^B, c^F]$

We wish to show that $c^0 \in [c^B, c^F]$, so that our imposition of $V_b = u'(c^0)$ is correct i.e. we can be sure that $V_b^0$ is bounded by $[V_b^F, V_b^B]$.

To start, note that we have defined

$$c^0 = x - g(d^0, a) \tag{16}$$

$$g_d(d^F, a) = \frac{V_a}{u'(c^F)} \tag{17}$$

$$g_d(d^B, a) = \frac{V_a}{u'(c^B)} \tag{18}$$

$$g_d(d^0, a) = \frac{V_a}{u'(c^0)} \tag{19}$$

where $x = r^b b + (1 - \xi)wz > 0$ stands in for liquid cash inflows. And also if we are calculating $c^0$, we know that optimal policies based on forward and backward liquid derivatives produced liquid drift counter to their assumed directions

$$\dot{b}^F = x - c^F - g(d^F, a) < 0 \tag{20}$$

$$\dot{b}^B = x - c^B - g(d^B, a) > 0 \tag{21}$$

The first step is to show $c^0 \geq c^B$.

1. To begin, suppose $c^0 < c^B$.

   (a) Combining their no–arbitrage equations (18 & 19) shows $d^0 < d^B$, by the increasing nature of $g_d(d, a)$

   $$g_d(d^0, a) = \frac{V_a}{u'(c^0)} < \frac{V_a}{u'(c^B)} = g_d(d^B, a)$$
   $$d^0 < d^B$$

   (b) Combining their drift equations (16 & 21) shows $d^0 > d^B$, by the increasing nature of $g(d, a)$

   $$x - g(d^0, a) < x - g(d^B, a) - \dot{b}^B$$
   $$g(d^0, a) > g(d^B, a)$$
   $$d^0 > d^B$$

This is a contradiction.

2. Now suppose $c^0 \geq c^B$.

   (a) Combining their no–arbitrage equations (18 & 19) shows $d^0 \geq d^B$, by the increasing nature of $g_d(d, a)$

$$g_d(d^0, a) = \frac{V_a}{u'(c^0)} \geq \frac{V_a}{u'(c^B)} = g_d(d^B, a)$$
$$d^0 \geq d^B$$

   (b) Combining their drift equations admits $d^0 \geq d^b$.

$$x - g(d^0, a) > x - g(d^B, a) - \dot{b}^B$$
$$g(d^0, a) < g(d^B, a) + \dot{b}^B$$

   Trying $d^0 \geq d^b$, and using the increasing nature of $g(d, a)$

$$g(d^B, a) \leq g(d^0, a) < g(d^B, a) + \dot{b}^B$$
$$0 \leq \dot{b}^B$$

   Which is established by equation (21).

   There is no contradiction here, so $c^0 \geq c^B$.

Equivalent steps establish $c^0 \leq c^F$, by symmetry. Hence $c^0 \in [c^B, c^F]$ QED.

**Lemma that** $d^0 \in [d^B, d^F]$ Given that $c^0 \in [c^B, c^F]$, for a given $V_a$ (assuming illiquid drift is not affected by $d$ choice), then we know

$$\frac{V_a}{u'(c^B)} \leq \frac{V_a}{u'(c^0)} \leq \frac{V_a}{u'(c^F)}$$

and therefore by the no–arbitrage FOC and increasing $g_d(d, a)$ in $d$, we know

$$d^B \leq d^0 \leq d^F$$

# References


Achdou, Y., J. Han, J.-M. Lasry, P.-L. Lions, and B. Moll (2022): "Income and wealth distribution in macroeconomics: A continuous-time approach," *The Review of Economic Studies*, 89, 45–86.

Barles, G. and P. E. Souganidis (1991): "Convergence of approximation schemes for fully nonlinear second order equations," *Asymptotic analysis*, 4, 271–283.

Kaplan, G., B. Moll, and G. L. Violante (2018): "Monetary policy according to HANK," *American Economic Review*, 108, 697–743.

Rendahl, P. (2022): "Continuous vs. discrete time: Some computational insights," *Journal of Economic Dynamics and Control*, 144, 104522.

Stokey, N. L. (1989): *Recursive methods in economic dynamics*, Harvard University Press.