Additional Codes using Finite Difference Method

Benjamin Moll

1 HJB Equation for Consumption-Saving Problem Without Uncertainty

Before considering the case with stochastic income in `http://www.princeton.edu/~moll/HACTproject/HACT_Numerical_Appendix.pdf`, it is useful to first really understand the case without uncertainty:

$$\rho v(a) = \max_c \; u(c) + v'(a)[w + ra - c] \tag{1}$$

with a state constraint $a \geq \underline{a}$, and we assume $r < \rho$ and $w > 0$ and $\underline{a} > -w/r$. For future reference denote by $s(a) = w + ra - c(a)$ where $c(a)$ is the optimal choice in (1). The state constraint implies $s(\underline{a}) = w + r\underline{a} - c(\underline{a}) \geq 0$. Since $u'(c(\underline{a})) = v'(\underline{a})$ and since u is concave therefore

$$v'(\underline{a}) \geq u'(w + r\underline{a}) \tag{2}$$

As above, we use a finite difference method and approximate the function v at J discrete points in the space dimension, $a_j, j = 1, ..., J$. We use equispaced grids, denote by Δa the distance between grid points, and use the short-hand notation $v_j \equiv v(a_j)$.

Again as before, one can implement either a so-called "explicit" method or an "implicit" method. As usual, the implicit method is the preferred approach because it is both more efficient and more stable/reliable. However, the explicit method is easier to explain so we turn to it first.

1.1 Explicit Method

Simplest Possible Algorithm. See matlab program `HJB_no_uncertainty_simple.m`. Given that there is no uncertainty and $r < \rho$, we know the following properties of the solution to (1): first, savings will be negative everywhere, $s(a) \leq 0$ all a; and the borrowing constraint will always bind and hence (2) holds with equality. Given these properties, an extremely simple algorithm can be used. In particular, use a backward difference approximation to v' everywhere

$$v_j' = \frac{v_j - v_{j-1}}{\Delta a}, \; j \geq 2, \quad v_1' = u'(w + ra_1) \tag{3}$$

and update the value function using

$$\frac{v_j^{n+1} - v_j^n}{\Delta} + \rho v_j^n = u(c_j^n) + (v_j^n)'[w + ra_j - c_j^n] \tag{4}$$

where $c_j^n = (u')^{-1}[(v_j^n)']$. As above Δ is the step size of the explicit scheme which cannot be too large (CFL condition). A small enough Δ also guarantees that the Barles-Souganidis conditions are satisfied. See `http://www.princeton.edu/~moll/HACT.pdf` and `http://www.princeton.edu/~moll/HACTproject/HACT_Numerical_Appendix.pdf` for more discussion.

Summary of Algorithm. Summarizing, the algorithm for finding a solution to the HJB equation (1) is as follows. Guess $v_j^0, j = 1, ..., J$ and for $n = 0, 1, 2, ...$ follow

1. Compute $(v_j^n)'$ from (3).
2. Compute c^n from $c_j^n = (u')^{-1}[(v_j^n)']$
3. Find v^{n+1} from (4).
4. If v^{n+1} is close enough to v^n: stop. Otherwise, go to step 1.

Upwind Scheme. Note that (3) is an "upwind scheme". As explained above, an upwind scheme uses a forward difference approximation whenever the drift of the state variable (here, savings $s_j^n = w + ra_j - c_j^n$) is positive and a backwards difference whenever it is negative. In the special case without uncertainty, we know that savings are negative everywhere and hence that one should always use the backwards difference approximation.

Instead of imposing that the backwards difference is always used, we could have let the upwind scheme "choose" the correct approximation as follows: first compute savings according to both the backwards and forward difference approximations $v'_{j,F}$ and $v'_{j,B}$

$$s_{j,F} = w + ra_j - (u')^{-1}(v'_{j,F}), \quad s_{j,B} = w + ra_j - (u')^{-1}(v'_{j,B})$$

where we suppress n superscripts for notational simplicity. Then use the following approximation for v'_j:

$$v'_j = v'_{j,F}\mathbf{1}_{\{s_{j,F}>0\}} + v'_{j,B}\mathbf{1}_{\{s_{j,B}<0\}} + \bar{v}'_j\mathbf{1}_{\{s_{j,F}<0<s_{j,B}\}} \tag{5}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function, and where $\bar{v}'_j = u'(w + ra_j)$. This scheme would find that $\mathbf{1}_{\{s_{j,B}<0\}}$ for all $j \geq 2$ and hence would pick the approximation in (3) by itself. This slightly more general solution algorithm is programmed up in `HJB_no_uncertainty_explicit.m`.

1.2 Implicit Method

See `HJB_no_uncertainty_implicit.m` and also see Section 1.2 of `http://www.princeton.edu/~moll/HACTproject/HACT_Numerical_Appendix.pdf` for a detailed explanation in the

version with uncertainty.

Relative to the explicit scheme in (4), an implicit differs in how v^n is updated. In particular, v^{n+1} is now implicitly defined by the equation

$$\frac{v_j^{n+1} - v_j^n}{\Delta} + \rho v_j^{n+1} = u(c_j^n) + (v_j^{n+1})'_F(w + ra_j - c_{j,F}^n)^+ + (v_j^{n+1})'_B(w + ra - c_{j,B}^n)^- \qquad (6)$$

where $c_j^n = (u')^{-1}[(v_j^n)']$ and $(v_j^n)'$ is given by (5). For any number x, the notation x^+ means "the positive part of x", i.e. $x^+ = \max\{x, 0\}$ and analogously $x^- = \min\{x, 0\}$, i.e. $[w+ra_j-c_{j,F}^n]^+ = \max\{w + ra_j - c_{j,F}^n, 0\}$ and $[w + ra_j - c_{j,B}^n]^- = \min\{w + ra_j - c_{j,B}^n, 0\}$.

Equation (6) constitutes a system of J linear equations, and it can be written in matrix notation using the following steps. Substituting the finite difference approximations to the derivatives, and defining $s_{j,F}^n = w + ra_j - c_{j,F}^n$ and similarly for $s_{j,B}^n$, (6) is

$$\frac{v_j^{n+1} - v_j^n}{\Delta} + \rho v_j^{n+1} = u(c_j^n) + \frac{v_{j+1}^{n+1} - v_j^{n+1}}{\Delta a}(s_{j,F}^n)^+ + \frac{v_j^{n+1} - v_{j-1}^{n+1}}{\Delta a}(s_{j,B}^n)^-$$

Collecting terms with the same subscripts on the right-hand side

$$\begin{aligned} &\frac{v_j^{n+1} - v_j^n}{\Delta} + \rho v_j^{n+1} = u(c_j^n) + v_{j-1}^{n+1}x_j + v_j^{n+1}y_j + v_{j+1}^{n+1}z_j \quad \text{where} \\ &x_j = -\frac{(s_{j,B}^n)^-}{\Delta a}, \quad y_j = -\frac{(s_{j,F}^n)^+}{\Delta a} + \frac{(s_{j,B}^n)^-}{\Delta a}, \quad z_j = \frac{(s_{j,F}^n)^+}{\Delta a} \end{aligned} \qquad (7)$$

Note that importantly $x_1 = z_J = 0$ so v_0^{n+1} and v_{J+1}^{n+1} are never used.

Equation (7) is a linear system which can be written in matrix notation as:

$$\frac{1}{\Delta}(v^{n+1} - v^n) + \rho v^{n+1} = u^n + \mathbf{A}^n v^{n+1}, \qquad \mathbf{A}^n = \begin{bmatrix} y_1 & z_1 & 0 & \cdots & 0 \\ x_2 & y_2 & z_2 & \ddots & 0 \\ 0 & x_3 & y_3 & z_3 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \ddots & \ddots & x_I & y_I \end{bmatrix}.$$

1.3 Results

Figure 1 plots the function $s(a)$.

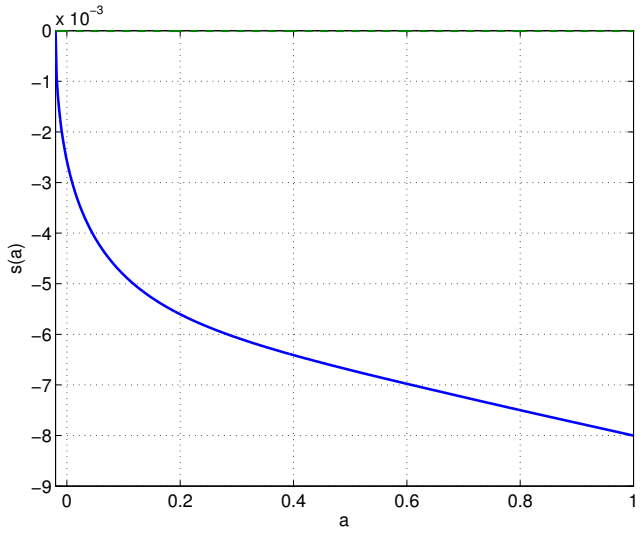

Figure 1: Savings Behavior in Model Without Uncertainty

2 Solving the Neoclassical Growth Model

See matlab codes `HJB_NGM.m` and `HJB_NGM_implicit`. Finally and for completeness, let us solve the neoclassical growth model which is the prototypical dynamic programming problem in macroeconomics. The HJB equation is

$$\rho V(k) = \max_c \; U(c) + V'(k)[F(k) - \delta k - c] \tag{8}$$

As before $s(k) = F(k) - \delta k - c(k)$ and $c(k) = (U')^{-1}(V'(k))$ denote optimal savings and consumption. We approximate V at I discrete grids points and use the short-hand notation $V_i = V(k_i)$. We first implement an explicit and then an implicit method. As usual, the implicit method is preferable due to better efficiency and stability properties.

2.1 Explicit Method

See `HJB_NGM.m`. The explicit method starts with a guess $V^0 = (V_1^0, ..., V_I^0)$ and for $n = 0, 1, 2, ...$ updates V according to

$$\frac{V_i^{n+1} - V_i^n}{\Delta} + \rho V_i^n = U(c_i^n) + (V_i^n)'[F(k_i) - \delta k_i - c_i^n] \tag{9}$$

$$c_i^n = (U')^{-1}[(V_i^n)'] \tag{10}$$

Upwind Scheme. The derivative $V'(k_i)$ is again approximated using an upwind scheme. That is compute savings according to both the backwards and forward difference approxima-

tions $V'_{i,F}$ and $V'_{i,B}$

$$s_{i,F} = F(k_i) - \delta k_i - (U')^{-1}(V'_{i,F}), \quad s_{i,B} = F(k_i) - \delta k_i - (U')^{-1}(V'_{i,B})$$

and then use the following approximation for V'_i:

$$V'_i = V'_{i,F}\mathbf{1}_{\{s_{i,F}>0\}} + V'_{i,B}\mathbf{1}_{\{s_{i,B}<0\}} + \bar{V}'_i\mathbf{1}_{\{s_{i,F}<0<s_{i,B}\}}$$

where $\bar{V}'_i = u'(F(k_i) - \delta k_i)$. Note again that the case $s_{i,F} > s_{i,B}$ will not occur because V is concave.

Remark. We know that the neoclassical growth model (8) has a steady state k^* satisfying $F'(k^*) = \rho + \delta$ and that at this steady state $V'(k^*) = U'(F(k^*) - \delta k^*)$. Note that the upwind scheme in effect uses the condition on the value function at the steady state k^* as a boundary condition. It then uses a backward difference approximation below the steady state, and a forward difference approximation above the steady state.

2.2 Implicit Method

See `HJB_NGM_implicit.m`. The algorithm is exactly the same as in Section 1.2. Also see Section 1.2 of `http://www.princeton.edu/~moll/HACTproject/HACT_Numerical_Appendix.pdf`.

2.3 Results.

Figure 2 plots the savings policy function in the neoclassical growth model.

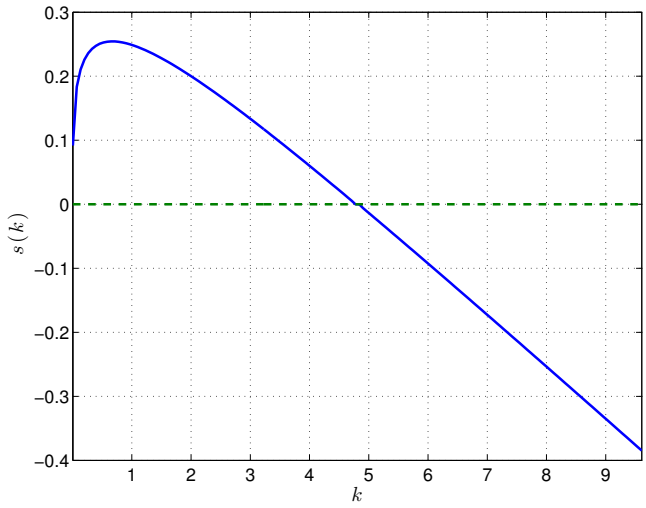

Figure 2: Savings Policy Function in Neoclassical Growth Model