Solving One-Sector Growth Model Using Implicit Finite Difference Method

Tatsuro Senga

The Economic Problem

- Continuous-time optimal growth problem:

$$\max_{c(t)} \int_0^\infty e^{-rt} \frac{c(t)^{1-s}}{1-s} dt$$

  subject to

$$\dot{k} = Ak^\alpha - \delta k - c$$

- Parameters:
  - r: discount rate
  - s: risk aversion
  - α: capital share
  - δ: depreciation rate
  - A: productivity

Hamilton-Jacobi-Bellman (HJB) Equation

- The HJB equation for this problem is:

$$rV(k) = \max_c \left\{ \frac{c^{1-s}}{1-s} + V'(k)(Ak^\alpha - \delta k - c) \right\}$$

- First-order condition for consumption:

$$c^{-s} = V'(k)$$

- Optimal consumption:

$$c = (V'(k))^{-1/s}$$

Discretization and Time Stepping

- The implicit scheme starts with:

$$\frac{V_i^{n+1} - V_i^n}{\Delta} + rV_i^{n+1} = u(c_i^n) + (V_i^{n+1})'(Ak_i^\alpha - \delta k_i - c_i^n)$$

where

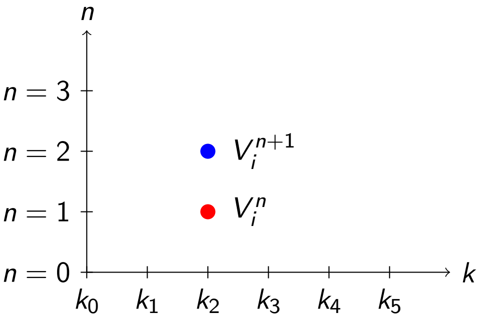
$$c_i^n = (V_i^n)'^{-1/s}$$

`V = (A*k.`$^a lpha).^(1-s)/(1-s)/rho; dV = zeros(I,1);$

Grid and Index Notation

- i denotes position in capital grid:
  - $i = 1, 2, \ldots, I$ where I is grid size
  - V_i is value at k_i
- n denotes time iteration:
  - $n = 0, 1, 2, \ldots$ until convergence
  - V^n is value function at iteration n
- Vector notation: $V^n = (V_1^n, V_2^n, \ldots, V_I^n)$

Grid and Time Structure



- V_i^n represents value at capital level i and iteration n
- Full solution: $I \times N$ matrix where N is final iteration

Capital Grid Construction

- First find steady state capital:

$$k_{ss} = \left(\frac{\alpha A}{r + \delta} \right)^{1/(1-\alpha)}$$

- Create capital grid:
  - $k_{min} = 0.001 k_{ss}$
  - $k_{max} = 2 k_{ss}$
  - $\Delta k = (k_{max} - k_{min})/(I - 1)$
  - $k_i = k_{min} + (i-1)\Delta k$

```
kss = (alpha*A/(rho+delta))^(1/(1-alpha));
kmin = 0.001*kss;
kmax = 2*kss;
k = linspace(kmin, kmax, I)';
dk = (kmax-kmin)/(I-1);
```

Value Function and Derivatives

Initial guess:

$$v_i^0 = \frac{(k_i^a)^{1-s}}{(1-s)r} \tag{1}$$

- Forward difference:

$$dV_{i,f} = (V_i^n)'_f = \frac{V_{i+1}^n - V_i^n}{\Delta k}$$

- Backward difference:

$$dV_{i,b} = (V_i^n)'_b = \frac{V_i^n - V_{i-1}^n}{\Delta k}$$

  - Boundary conditions:
    - At k_{min}: $V'(k_{min}) = (Ak_{min}^{\alpha} - \delta k_{min})^{-s}$
    - At k_{max}: $V'(k_{max}) = (Ak_{max}^{\alpha} - \delta k_{max})^{-s}$

```
tv = (k.**a).**(1-s)/(1-s)/r;
dVf(1:I-1) = diff(v)/dk;
dVf(I) = (kmax**a - d*kmax)**(-s);
dVb(2:I) = diff(v)/dk;
dVb(1) = (kmin**a - d*kmin)**(-s);
```

Consumption and Drift Terms

Consumption from derivatives:

$$c_{i,f} = (dV_{i,f})^{-1/s} \tag{5}$$

$$c_{i,b} = (dV_{i,b})^{-1/s} \tag{6}$$

Drift terms:

$$\mu_f = k^a - dk - c_{i,f} \tag{7}$$

$$\mu_b = k^a - dk - c_{i,b} \tag{8}$$

```
cf = dVf.**(-1/s);
muf = k.**a - d.*k - cf;
cb = dVb.**(-1/s);
mub = k.**a - d.*k - cb;
```

Upwind Scheme

Choose derivative based on drift:

$$V'(k) = \begin{cases} V_f'(k) & \text{if } \mu(k) > 0 \\ V_b'(k) & \text{if } \mu(k) < 0 \\ V_0'(k) & \text{if } \mu(k) = 0 \end{cases}$$

$$dV_{upwind} = dV_f \cdot I_f + dV_b \cdot I_b + dV_0 \cdot I_0 \tag{9}$$

where:

$$I_f = [\mu_f > 0] \tag{10}$$

$$I_b = [\mu_b < 0] \tag{11}$$

$$I_0 = 1 - I_f - I_b \tag{12}$$

```
If = muf > 0;
Ib = mub < 0;
I0 = (1-If-Ib);
```

$dV_{Upwind} = dVf. * If + dVb. * Ib + dV0. * I0;$

Matrix System Construction: Step 1

Starting with the discretized HJB equation:

$$\frac{V_i^{n+1} - V_i^n}{\Delta} + rV_i^{n+1} = u(c_i^n) + (V_i^{n+1})'(k_i^\alpha - \delta k_i - c_i^n) \tag{1}$$

After applying upwind scheme:

$$\begin{aligned}\frac{V_i^{n+1} - V_i^n}{\Delta} + rV_i^{n+1} = u(c_i^n) &+ (V_{i,F}^{n+1})'(k_i^\alpha - \delta k_i - c_i^n)^+ \\ &+ (V_{i,B}^{n+1})'(k_i^\alpha - \delta k_i - c_i^n)^-\end{aligned} \tag{2}$$

Matrix System Construction: Step 2

Substituting forward and backward finite differences and collecting terms for V_i^{n+1} on the RHS:

$$\frac{V_i^{n+1} - V_i^n}{\Delta} + rV_i^{n+1} = u(c_i^n) + x_i V_{i-1}^{n+1} + y_i V_i^{n+1} + z_i V_{i+1}^{n+1} \tag{8}$$

Define coefficients:

$$x_i = -\min(\mu_b, 0)/\Delta k \tag{5}$$
$$y_i = -\max(\mu_f, 0)/\Delta k + \min(\mu_b, 0)/\Delta k \tag{6}$$
$$z_i = \max(\mu_f, 0)/\Delta k \tag{7}$$

```
X = -min(mub,0)/dk;
Y = -max(muf,0)/dk + min(mub,0)/dk;
Z = max(muf,0)/dk;
```

Matrix System Construction: Step 3

Forms tridiagonal matrix P^n:

$$A^n = \begin{bmatrix} y_1 & z_1 & 0 & \cdots & 0 \\ x_2 & y_2 & z_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & x_{I-1} & y_{I-1} & z_{I-1} \\ 0 & \cdots & 0 & x_I & y_I \end{bmatrix} \tag{9}$$

```
A = spdiags(Y,0,I,I) + ... spdiags(X(2:I),-1,I,I) + ... spdiags([0;Z(1:I-1)
```

Matrix System

Rearranging equation (8):

$$(\frac{1}{\Delta} + r)V_i^{n+1} - (x_i V_{i-1}^{n+1} + y_i V_i^{n+1} + z_i V_{i+1}^{n+1}) = u(c_i^n) + \frac{V_i^n}{\Delta} \tag{10}$$

In matrix form:

$$[(\frac{1}{\Delta} + r)I - A^n]V^{n+1} = U^n + \frac{V^n}{\Delta} \tag{11}$$

```
X = -min(mub,0)/dk;
Y = -max(muf,0)/dk + min(mub,0)/dk;
Z = max(muf,0)/dk;
A = spdiags(Y,0,I,I) + spdiags(X(2:I),-1,I,I) + ... spdiags([0;Z(1:I-1)],1,
```

Solving the Linear System

Matrix equation to solve:

$$Bv^{n+1} = b \tag{16}$$

where:

- $B = (r + \frac{1}{\Delta})I - A$
- $b = u + \frac{v^n}{\Delta}$

```
B = (r + 1/Delta)*speye(I) - A;
b = u + v/Delta;
```

The command `tv = B\b` in MATLAB to obtain v.

Solution Algorithm

- Main iteration steps:
  1. Compute derivatives $(V_i^n)'_f$, $(V_i^n)'_b$
  2. Calculate consumption c_f^n, c_b^n
  3. Evaluate drift terms μ_f, μ_b
  4. Form matrix B and vector b
  5. Solve $BV^{n+1} = b$
  6. Check convergence: $\|V^{n+1} - V^n\| < \epsilon$