Lecture 3:

Hamilton-Jacobi-Bellman Equations

Distributional Macroeconomics

Part II of ECON 2149

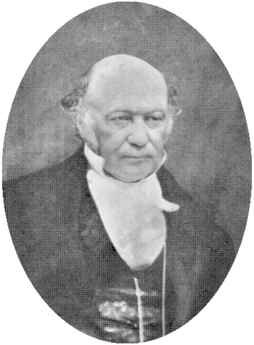
Benjamin Moll

Harvard University, Spring 2018

Outline

1. Hamilton-Jacobi-Bellman equations in deterministic settings
2. Numerical solution: finite difference method

Hamilton-Jacobi-Bellman Equation: Some "History"



(a) William Hamilton

(b) Carl Jacobi

(c) Richard Bellman

- Aside: why called "dynamic programming"?
- Bellman: *"Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities."* http://en.wikipedia.org/wiki/Dynamic_programming#History

Hamilton-Jacobi-Bellman Equations

- Recall the generic deterministic optimal control problem from Lecture 1:

$$v(x_0) = \max_{\{\alpha(t)\}_{t\geq 0}} \int_0^\infty e^{-\rho t} r(x(t), \alpha(t))\, dt$$

subject to the law of motion for the state

$$\dot{x}(t) = f(x(t), \alpha(t)) \quad \text{and} \quad \alpha(t) \in A$$

for $t \geq 0$, $x(0) = x_0$ given.

- $\rho \geq 0$: discount rate
- $x \in X \subseteq \mathbb{R}^N$: state vector
- $\alpha \in A \subseteq \mathbb{R}^M$: control vector
- $r : X \times A \to \mathbb{R}$: instantaneous return function

Example: Neoclassical Growth Model

$$v(k_0) = \max_{\{c(t)\}_{t\geq 0}} \int_0^\infty e^{-\rho t} u(c(t)) dt$$

subject to

$$\dot{k}(t) = F(k(t)) - \delta k(t) - c(t)$$

for $t \geq 0$, $k(0) = k_0$ given.

- Here the state is $x = k$ and the control $\alpha = c$
- $r(x, \alpha) = u(\alpha)$
- $f(x, \alpha) = F(x) - \delta x - \alpha$

Generic HJB Equation

- How to analyze these optimal control problems? Here: "cookbook approach"

- Result: the value function of the generic optimal control problem satisfies the Hamilton-Jacobi-Bellman equation

$$\rho v(x) = \max_{\alpha \in A} \; r(x, \alpha) + v'(x) \cdot f(x, \alpha)$$

- In the case with more than one state variable $N > 1$, $v'(x) \in \mathbb{R}^N$ is the gradient of the value function.

Example: Neoclassical Growth Model

- "cookbook" implies:

$$\rho v(k) = \max_{c} \; u(c) + v'(k)(F(k) - \delta k - c)$$

- Proceed by taking first-order conditions etc

$$u'(c) = v'(k)$$

Derivation from Discrete-time Bellman

- Here: derivation for neoclassical growth model
- Extra class notes: generic derivation
- Time periods of length Δ
- discount factor

$$\beta(\Delta) = e^{-\rho\Delta}$$

- Note that $\lim_{\Delta\to 0} \beta(\Delta) = 1$ and $\lim_{\Delta\to\infty} \beta(\Delta) = 0$
- Discrete-time Bellman equation:

$$v(k_t) = \max_{c_t} \ \Delta u(c_t) + e^{-\rho\Delta} v(k_{t+\Delta}) \quad \text{s.t.}$$

$$k_{t+\Delta} = \Delta(F(k_t) - \delta k_t - c_t) + k_t$$

Derivation from Discrete-time Bellman

- For small Δ (will take $\Delta \to 0$), $e^{-\rho\Delta} = 1 - \rho\Delta$

$$v(k_t) = \max_{c_t} \; \Delta u(c_t) + (1 - \rho\Delta) v(k_{t+\Delta})$$

- Subtract $(1 - \rho\Delta) v(k_t)$ from both sides

$$\rho\Delta v(k_t) = \max_{c_t} \; \Delta u(c_t) + (1 - \Delta\rho)(v(k_{t+\Delta}) - v(k_t))$$

- Divide by Δ and manipulate last term

$$\rho v(k_t) = \max_{c_t} \; u(c_t) + (1 - \Delta\rho) \frac{v(k_{t+\Delta}) - v(k_t)}{k_{t+\Delta} - k_t} \frac{k_{t+\Delta} - k_t}{\Delta}$$

- Take $\Delta \to 0$

$$\rho v(k_t) = \max_{c_t} \; u(c_t) + v'(k_t)\dot{k}_t$$

Connection Between HJB Equation and Hamiltonian

- Hamiltonian

$$\mathcal{H}(x, \alpha, \lambda) = r(x, \alpha) + \lambda f(x, \alpha)$$

- HJB equation

$$\rho v(x) = \max_{\alpha \in A} \; r(x, \alpha) + v'(x) f(x, \alpha)$$

- Connection: $\lambda(t) = v'(x(t))$, i.e. co-state = shadow value
- HJB can be written as $\rho v(x) = \max_{\alpha \in A} \mathcal{H}(x, \alpha, v'(x))$ ...
- ... hence the “Hamilton” in Hamilton-Jacobi-Bellman
- Can show: playing around with FOC and envelope condition gives conditions for optimum from Lecture 1
- Mathematicians’ notation: in terms of maximized Hamiltonian H

$$\rho v(x) = H(x, v'(x))$$

$$H(x, p) := \max_{\alpha \in A} \; r(x, \alpha) + p f(x, \alpha)$$

Some general, somewhat philosophical thoughts

- MAT 101 way ("first-order ODE needs one boundary condition") is not the right way to think about HJB equations
- these equations have very special structure which one should exploit when analyzing and solving them
- Particularly true for computations
- Important: all results/algorithms apply to problems with more than one state variable, i.e. it doesn't matter whether you solve ODEs or PDEs

Existence and Uniqueness of Solutions to (HJB)

Recall Hamilton-Jacobi-Bellman equation:

$$\rho v(x) = \max_{\alpha \in A} \left\{ r(x, \alpha) + v'(x) \cdot f(x, \alpha) \right\} \qquad \text{(HJB)}$$

Two key results, analogous to discrete time:

- Theorem 1 (HJB) has a unique "nice" solution
- Theorem 2 "nice" solution equals value function, i.e. solution to "sequence problem"
- Here: "nice" solution = "viscosity solution"
- See supplement "Viscosity Solutions for Dummies" `http://www.princeton.edu/~moll/viscosity_slides.pdf`
- Theorems 1 and 2 hold for both ODE and PDE cases, i.e. also with multiple state variables...
- ... also hold if value function has kinks (e.g. from non-convexities)
- Remark re Thm 1: in typical application, only very weak boundary conditions needed for uniqueness ($\leq$'s, boundedness assumption)

Numerical Solution of HJB Equations

One-Slide Summary of Numerical Method

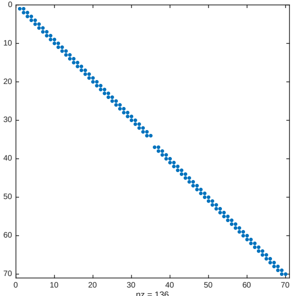
- Consider general HJB equation:

$$\rho v(x) = \max_{\alpha} \; r(x,\alpha) + v'(x) \cdot f(x,\alpha)$$

- Will discretize and solve using finite difference method
- Discretization $\Rightarrow$ system of non-linear equations

$$\rho \mathbf{v} = \mathbf{r}(\mathbf{v}) + \mathbf{A}(\mathbf{v})\mathbf{v}$$

where **A** is a sparse (tri-diagonal) transition matrix



Barles-Souganidis

- There is a well-developed theory for numerical solution of HJB equation using finite difference methods
- Key paper: Barles and Souganidis (1991), "Convergence of approximation schemes for fully nonlinear second order equations
  `https://www.dropbox.com/s/vhw5qqrczw3dvw3/barles-souganidis.pdf?dl=0`
- Result: finite difference scheme "converges" to unique viscosity solution under three conditions
  1. monotonicity
  2. consistency
  3. stability
- Good reference: Tourin (2013), "An Introduction to Finite Difference Methods for PDEs in Finance."

Problem we will work with: neoclassical growth model

- Explain using neoclassical growth model, easily generalized to other applications

$$\rho v(k) = \max_c \; u(c) + v'(k)(F(k) - \delta k - c)$$

- Functional forms

$$u(c) = \frac{c^{1-\sigma}}{1-\sigma}, \quad F(k) = k^{\alpha}$$

- Use finite difference method
    - Two MATLAB codes

    http://www.princeton.edu/~moll/HACTproject/HJB_NGM.m

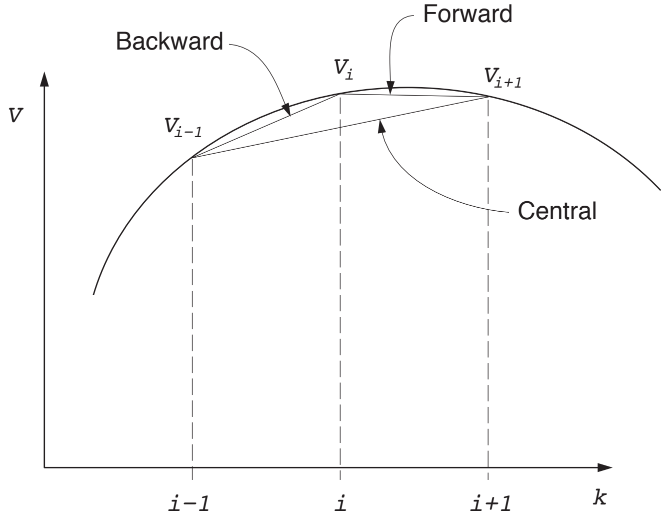
    http://www.princeton.edu/~moll/HACTproject/HJB_NGM_implicit.m

Finite Difference Approximations to $v'(k_i)$

- Approximate $v(k)$ at I discrete points in the state space, k_i, $i = 1, ..., I$. Denote distance between grid points by Δk.
- Shorthand notation
$$v_i = v(k_i)$$
- Need to approximate $v'(k_i)$.
- Three different possibilities:

$$v'(k_i) \approx \frac{v_i - v_{i-1}}{\Delta k} = v'_{i,B} \qquad \text{backward difference}$$
$$v'(k_i) \approx \frac{v_{i+1} - v_i}{\Delta k} = v'_{i,F} \qquad \text{forward difference}$$
$$v'(k_i) \approx \frac{v_{i+1} - v_{i-1}}{2\Delta k} = v'_{i,C} \qquad \text{central difference}$$

Finite Difference Approximations to $v'(k_i)$



Finite Difference Approximation

FD approximation to HJB is

$$\rho v_i = u(c_i) + v_i' \times (F(k_i) - \delta k_i - c_i) \qquad (*)$$

where $c_i = (u')^{-1}(v_i')$, and v_i' is one of backward, forward, central FD approximations.

Two complications:

1. which FD approximation to use? "Upwind scheme"
2. $(*)$ is extremely non-linear, need to solve iteratively: "explicit" vs. "implicit method"

My strategy for next few slides:

- what works
- at end of lecture: why it works (Barles-Souganidis)

Which FD Approximation?

- Which of these you use is extremely important
- Best solution: use so-called "upwind scheme." Rough idea:
  - forward difference whenever drift of state variable positive
  - backward difference whenever drift of state variable negative
- In our example: define

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

- Approximate derivative as follows

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

  where $\mathbf{1}_{\{\cdot\}}$ is indicator function, and $\bar{v}'_i = u'(F(k_i) - \delta k_i)$.
- Where does $\bar{v}'_i$ term come from? Answer:
  - since v is concave, $v'_{i,F} < v'_{i,B}$ (see figure) $\Rightarrow s_{i,F} < s_{i,B}$
  - if $s'_{i,F} < 0 < s'_{i,B}$, set $s_i = 0 \Rightarrow v'(k_i) = u'(F(k_i) - \delta k_i)$, i.e. we're at a steady state.

Sparsity

- Recall discretized HJB equation

$$\rho v_i = u(c_i) + v_i' \times (F(k_i) - \delta k_i - c_i), \quad i = 1, ..., I$$

- This can be written as

$$\rho v_i = u(c_i) + \frac{v_{i+1} - v_i}{\Delta k} s_{i,F}^{+} + \frac{v_i - v_{i-1}}{\Delta k} s_{i,B}^{-}, \quad i = 1, ..., I$$

  Notation: for any x, $x^+ = \max\{x, 0\}$ and $x^- = \min\{x, 0\}$

- Can write this in matrix notation

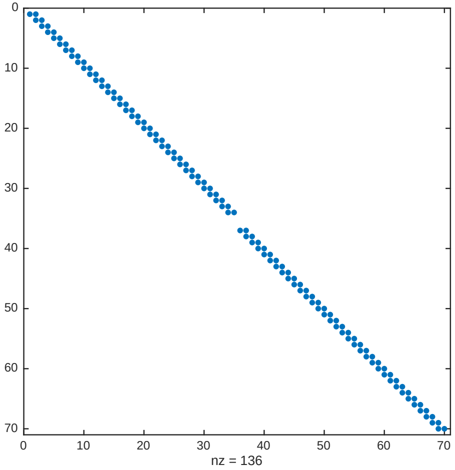
$$\rho v_i = u(c_i) + \begin{bmatrix} -\frac{s_{i,B}^{-}}{\Delta k} & \frac{s_{i,B}^{-}}{\Delta k} - \frac{s_{i,F}^{+}}{\Delta k} & \frac{s_{i,F}^{+}}{\Delta k} \end{bmatrix} \begin{bmatrix} v_{i-1} \\ v_i \\ v_{i+1} \end{bmatrix}$$

  and hence

$$\rho \mathbf{v} = \mathbf{u} + \mathbf{A}\mathbf{v}$$

  where $\mathbf{A}$ is $I \times I$ (I= no of grid points) and looks like...

Visualization of **A** (output of `spy(A)` in Matlab)



The matrix **A**

- FD method approximates process for k with discrete Poisson process, **A** summarizes Poisson intensities
  - entries in row i:

$$\begin{bmatrix} \underbrace{-\frac{s_{i,B}^{-}}{\Delta k}}_{\text{inflow}_{i-1}\geq 0} & \underbrace{\frac{s_{i,B}^{-}}{\Delta k} - \frac{s_{i,F}^{+}}{\Delta k}}_{\text{outflow}_{i}\leq 0} & \underbrace{\frac{s_{i,F}^{+}}{\Delta k}}_{\text{inflow}_{i+1}\geq 0} \end{bmatrix} \begin{bmatrix} v_{i-1} \\ v_i \\ v_{i+1} \end{bmatrix}$$

  - negative diagonals, positive off-diagonals, rows sum to zero:
  - tridiagonal matrix, very sparse

- **A** (and **u**) depend on **v** (nonlinear problem)

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

- Next: iterative method...

Iterative Method

- Idea: Solve FOC for given $\mathbf{v}^n$, update $\mathbf{v}^{n+1}$ according to

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^n = u(c_i^n) + (v^n)'(k_i)(F(k_i) - \delta k_i - c_i^n) \qquad (*)$$

- Algorithm: Guess $v_i^0, i = 1, ..., I$ and for $n = 0, 1, 2, ...$ follow
  1. Compute $(v^n)'(k_i)$ using FD approx. on previous slide.
  2. Compute c^n from $c_i^n = (u')^{-1}[(v^n)'(k_i)]$
  3. Find $\mathbf{v}^{n+1}$ from $(*)$.
  4. If $\mathbf{v}^{n+1}$ is close enough to $\mathbf{v}^n$: stop. Otherwise, go to step 1.

- See `http://www.princeton.edu/~moll/HACTproject/HJB_NGM.m`

- Important parameter: Δ = step size, cannot be too large ("CFL condition").

- Pretty inefficient: I need 5,990 iterations (though quite fast)

Efficiency: Implicit Method

- Efficiency can be improved by using an "implicit method"

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^{n+1} = u(c_i^n) + (v_i^{n+1})'(k_i)[F(k_i) - \delta k_i - c_i^n]$$

- Each step n involves solving a linear system of the form

$$\frac{1}{\Delta}(\mathbf{v}^{n+1} - \mathbf{v}^n) + \rho \mathbf{v}^{n+1} = \mathbf{u} + \mathbf{A}_n \mathbf{v}^{n+1}$$
$$\left((\rho + \tfrac{1}{\Delta})\mathbf{I} - \mathbf{A}_n\right) \mathbf{v}^{n+1} = \mathbf{u} + \tfrac{1}{\Delta}\mathbf{v}^n$$

- but $\mathbf{A}_n$ is super sparse $\Rightarrow$ super fast
- See `http://www.princeton.edu/~moll/HACTproject/HJB_NGM_implicit.m`
- In general: implicit method preferable over explicit method
    1. stable regardless of step size Δ
    2. need much fewer iterations
    3. can handle many more grid points

Implicit Method: Practical Consideration

- In Matlab, need to explicitly construct **A** as sparse to take advantage of speed gains
- Code has part that looks as follows

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

- Constructing full matrix – slow

```
for i=2:I-1
    A(i,i-1) = X(i);
    A(i,i) = Y(i);
    A(i,i+1) = Z(i);
end
A(1,1)=Y(1); A(1,2) = Z(1);
A(I,I)=Y(I); A(I,I-1) = X(I);
```

- Constructing sparse matrix – fast

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

Just so you remember: one-slide summary again

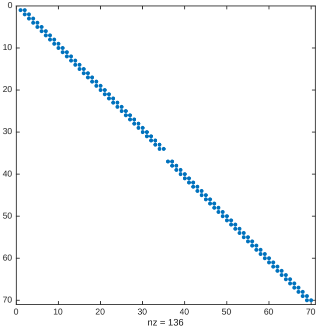
- Consider general HJB equation:

$$\rho v(x) = \max_{\alpha} \; r(x, \alpha) + v'(x) \cdot f(x, \alpha)$$

- Discretization $\Rightarrow$ system of non-linear equations

$$\rho \mathbf{v} = \mathbf{r}(\mathbf{v}) + \mathbf{A}(\mathbf{v})\mathbf{v}$$

where $\mathbf{A}$ is a sparse (tri-diagonal) transition matrix



Non-Convexities

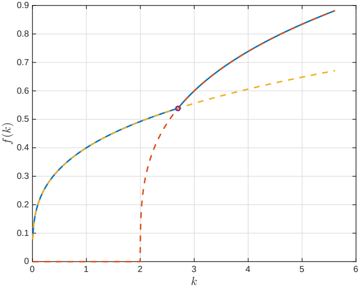
Non-Convexities

- Consider growth model

$$\rho v(k) = \max_c \ u(c) + v'(k)(F(k) - \delta k - c).$$

- But drop assumption that F is strictly concave. Instead: "butterfly"

$$F(k) = \max\{F_L(k), F_H(k)\},$$
$$F_L(k) = A_L k^\alpha,$$
$$F_H(k) = A_H((k-\kappa)^+)^\alpha, \quad \kappa > 0, \ A_H > A_L$$

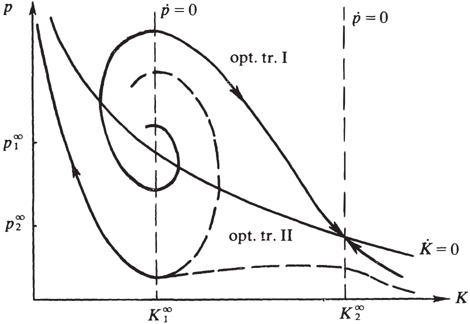


Standard Methods

- Discrete time: first-order conditions

$$u'(F(k) - \delta k - k') = \beta v'(k')$$

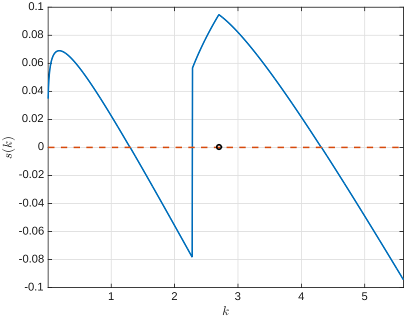
  no longer sufficient, typically multiple solutions
  - some applications: sidestep with lotteries (Prescott-Townsend)
- Continuous time: Skiba (1978)



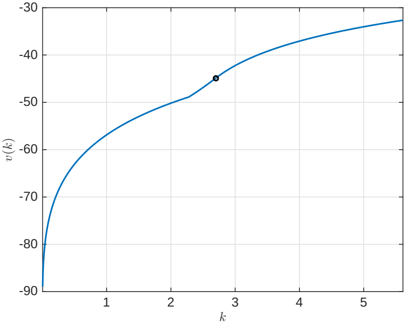
Instead: Using Finite-Difference Scheme

Nothing changes, use same exact algorithm as for growth model with concave production function

`http://www.princeton.edu/~moll/HACTproject/HJB_NGM_skiba.m`

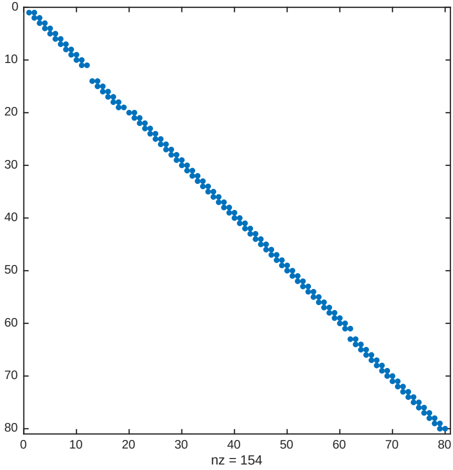


(a) Saving Policy Function



(b) Value Function

Visualization of **A** (output of `spy(A)` in Matlab)



Appendix

Why this works? Barles-Souganidis

- Here: version with one state variable, but generalizes
- Can write any HJB equation with one state variable as

$$0 = G(k, v(k), v'(k), v''(k)) \qquad \text{(G)}$$

- Corresponding FD scheme

$$0 = S\left(\Delta k, k_i, v_i; v_{i-1}, v_{i+1}\right) \qquad \text{(S)}$$

- Growth model

$$G(k, v(k), v'(k), v''(k)) = \rho v(k) - \max_c \; u(c) + v'(k)(F(k) - \delta k - c)$$

$$S\left(\Delta k, k_i, v_i; v_{i-1}, v_{i+1}\right) = \rho v_i - u(c_i) - \frac{v_{i+1} - v_i}{\Delta k}(F(k_i) - \delta k_i - c_i)^+ - \frac{v_i - v_{i-1}}{\Delta k}(F(k_i) - \delta k_i - c_i)^-$$

Why this works? Barles-Souganidis

1. Monotonicity: the numerical scheme is monotone, that is S is non-increasing in both v_{i-1} and v_{i+1}

2. Consistency: the numerical scheme is consistent, that is for every smooth function v with bounded derivatives

$$S\left(\Delta k, k_i, v(k_i); v(k_{i-1}), v(k_{i+1})\right) \to G(v(k), v'(k), v''(k))$$

   as $\Delta k \to 0$ and $k_i \to k$.

3. Stability: the numerical scheme is stable, that is for every $\Delta k > 0$, it has a solution v_i, $i = 1, .., I$ which is uniformly bounded independently of Δk.

Why this works? Barles-Souganidis

Theorem (Barles-Souganidis)

If the scheme satisfies the monotonicity, consistency and stability conditions 1 to 3, then as $\Delta k \to 0$ *its solution* $v_i, i = 1, ..., I$ *converges locally uniformly to the unique viscosity solution of* (G)

- Note: "convergence" here has nothing to do with iterative algorithm converging to fixed point
- Instead: convergence of v_i as $\Delta k \to 0$. More momentarily.

Intuition for Monotonicity

- Write (S) as

$$\rho v_i = \tilde{S}(\Delta k, k_i, v_i; v_{i-1}, v_{i+1})$$

- For example, in growth model

$$\begin{aligned}\tilde{S}(\Delta k, k_i, v_i; v_{i-1}, v_{i+1}) = u(c_i) &+ \frac{v_{i+1} - v_i}{\Delta k}(F(k_i) - \delta k_i - c_i)^+ \\ &+ \frac{v_i - v_{i-1}}{\Delta k}(F(k_i) - \delta k_i - c_i)^-\end{aligned}$$

- Monotonicity: $\tilde{S} \uparrow$ in v_{i-1}, v_{i+1} ($\Leftrightarrow S \downarrow$ in v_{i-1}, v_{i+1})
- Intuition: if my continuation value at $i-1$ or $i+1$ is larger, I must be at least as well off (i.e. v_i on LHS must be at least as high)

Checking the Monotonicity Condition in Growth Model

- Recall upwind scheme:

$$S\left(\Delta k, k_i, v_i; v_{i-1}, v_{i+1}\right) = \rho v_i - u(c_i) - \frac{v_{i+1} - v_i}{\Delta k}(F(k_i) - \delta k_i - c_i)^+ \\ - \frac{v_i - v_{i-1}}{\Delta k}(F(k_i) - \delta k_i - c_i)^-$$

- Can check: satisfies monotonicity: S is indeed non-increasing in both v_{i-1} and v_{i+1}

- c_i depends on v_i's but doesn't affect monotonicity due to envelope condition

Meaning of "Convergence"

Convergence is about $\Delta k \to 0$. What, then, is content of theorem?

- have a system of I non-linear equations $S(\Delta k, k, v_i; v_{i-1}, v_{i+1}) = 0$
- need to solve it somehow
- Theorem guarantees that solution (for given Δk) converges to solution of the HJB equation (G) as Δk.

Why does iterative scheme work? Two interpretations:

1. Newton method for solving system of non-linear equations (S)
2. Iterative scheme $\Leftrightarrow$ solve (HJB) backward in time

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^n = u(c_i^n) + (v^n)'(k_i)(F(k_i) - \delta k_i - c_i^n)$$

in effect sets $v(k, T) =$ initial guess and solves

$$\rho v(k,t) = \max_c \; u(c) + \partial_k v(k,t)(F(k) - \delta k - c) + \partial_t v(k,t)$$

backwards in time. $v(k) = \lim_{t \to -\infty} v(k,t)$.