Heterogeneous-Agent Models and Methods

Benjamin Moll
London School of Economics

Slides at `https://benjaminmoll.com/STEG_course/`

STEG Course "Key Concepts in Macro Development"

Why Heterogeneous-Agent Models? Richard's Take:

Frictions and Heterogeneity in Modern Macro

Major focus of macro over the last 20 years has been the development of models that incorporate rich specifications of heterogeneity and "frictions" that can simultaneously can speak to aggregate outcomes while also addressing a rich set of cross-sectional facts.

These models present an opportunity for a unified analysis of micro and macro development within the same framework. Recent paper by Buera, Kaboski and Townsend is a great overview of this agenda.

STEG seeks to actively promote this agenda and further facilitate interaction between individuals across groups.

Progress will come from both bottom-up and top-down approaches.

My goal: enable you to work with models like...

1. Buera, Kaboski and Townsend (2021), “From Micro to Macro Development” `https://www.nber.org/papers/w28423`

2. Banerjee and Duflo (2005), “Growth Theory Through the Lens of Development Economics”

   `https://www.sciencedirect.com/science/article/pii/S1574068405010075`

3. Cavalcanti, Kaboski, Martins and Santos (2021), “Dispersion in Financing Costs and Development” `https://www.nber.org/papers/w28635`

4. Buera and Shin (2013) “Financial Frictions and the Persistence of History” `https://www.jstor.org/stable/10.1086/670271?seq=1`

5. Lagakos, Mobarak & Waugh, “The Welfare Effects of Encouraging Rural-Urban Migration” `https://www.nber.org/papers/w24193`

An excerpt from Banerjee-Duflo (2005)

6. Towards a non-aggregative growth theory

6.1. An illustration

The presumption of neo-classical growth theory was that being a citizen of a poor country gives one access to many exciting investment opportunities, which eventually lead on to convergence. The point of the previous section was to argue that most citizens of poor countries are not in a position to enjoy most of these opportunities, either because markets do not do what they ought to or the government does what it ought not to, or because people find it psychologically difficult to do what is expected of them.

What can we say about the long-run evolution of an economy where there are rewarding opportunities that are not necessarily exploited? In this section we will explore this question under the assumption that the only source of inefficiency in this economy comes from limited access to credit. The goal is to illustrate what non-aggregative growth theory might look like, rather than to suggest an alternative canonical model.

The model we have in mind is as follows: There are individual production functions associated with every participant in this economy that are assumed to be identical and a function of capital alone ($F(K)$) but otherwise quite general. In particular, we do assume that they are concave. Individuals maximize an intertemporal utility function of the form:

$$\sum_{t=0}^{\infty} \delta^t U(C_t), \quad 0 < \delta < 1,$$

$$U(C_t) = \frac{c^{1-\phi}}{1-\phi}, \quad \phi > 0.$$

An excerpt from Banerjee-Duflo (2005)

Finally we consider the case of "S-shaped" production functions, which are production functions that are initially convex and then concave. The Cobb–Douglas with an initial set-up cost discussed at length in Section 5.2 is a special case of this kind of technology.

What happens in the long run in this model depends on the initial distribution of income. When the distribution is such that most people in the economy can afford to invest in the concave part of the production function, the economy converges to a situation that is isomorphic to the diminishing returns case, with the entire population "escaping" the convex region of the production function.

The more unusual case is the one where some people start too poor to invest in the concave region of the production function. The poorer among such people will earn very low returns if they were to invest and therefore will prefer to be lenders. Now, as long as the interest rate on savings is less than $1/\delta$, they will decumulate capital (since the interest is less than the discount factor) and eventually their wealth will go to zero. On the other hand, anyone in this economy who started rich enough to want to borrow will stay rich, even though they are also dissaving, in part because at the same time they benefit from the low interest rates. The economy will converge to a steady state where the interest rate is $1/\delta$, those who started rich continue to be rich and those who started poor remain poor (in fact have zero wealth).

This is classic poverty trap: Moreover, since no one escapes from poverty, nor falls into it, there is a continuum of such poverty traps in this model. This kind of multiplicity is, however, fragile with respect to the introduction of random shocks that allow some of the poor to escape poverty and impoverish some of the rich.

My point

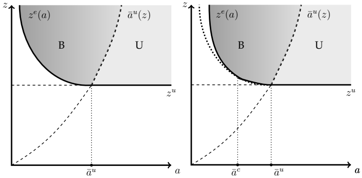
- This "non-aggregative growth theory" is exactly heterogeneous-agent macro
- B&D's chapter: speculative, verbal discussion
- Here instead teach you how to solve and analyze such models
    - I (for one) am not smart enough to figure out how models work without solving them!

Cavalcanti, Kaboski, Martins and Santos (2021)

ABSTRACT

Most aggregate theories of financial frictions model credit available at a single cost of financing but rationed. However, using a comprehensive firm-level credit registry, we document both high levels and high dispersion in credit spreads to Brazilian firms. We develop a quantitative dynamic general equilibrium model in which dispersion in spreads arises from intermediation costs and market power. Calibrating to the Brazilian data, we show that, for equivalent levels of external financing, dispersion has more profound impacts on aggregate development than single-price credit rationing and yields firm dynamics that are more consistent with observed patterns.

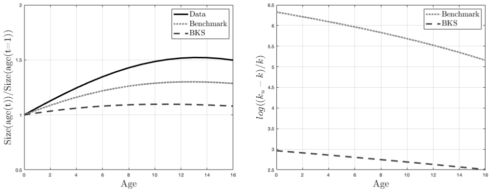
Figure 1: Occupational choice, $\chi = 0$.



(a) Full enforcement, $\phi = 1$. (b) Imperfect enforcement, $\phi < 1$.

Notes: The light gray shaded area, **U**, contains the measure of unconstrained entrepreneurs. The dark gray shaded area, **B**, displays the measure of constrained borrowers. The white area below the curve $z^e(a)$ represents the measure of workers.

Figure 2: Firm Dynamics



(a) Relative firm size by age. (b) $\log((k_u - k)/k)$ by age.

- Paper uses continuous-time methods I will teach you today
- Tiago, Joe, Bruno & Cezar were kind enough to share code so you can play around with it yourself! `http://benjaminmoll.com/ckms_code/`
  (Note: .zip file, my Google Chrome tries to block download)

Outline

1. Resources for discrete-time heterogeneous-agent models
2. Why continuous time?
3. Continuous-time Bellman (HJB) equations
4. Textbook heterogeneous-agent model
5. Numerical solution of HJB equations
6. Numerical solution of textbook heterogeneous-agent model
7. Problems with non-convexities
    - capital accumulation w S-shaped production functions (Skiba)
    - occupational choice (Cavalcanti-Kaboski-Martins-Santos)
    - ...

Way too much material for 75 mins! $\Rightarrow$ skip slides saying "(skip)" on top

Background materials I haven't mentioned yet

- Achdou et al (2020) "Income and Wealth Distribution in Macro: A Continuous-Time Approach" `https://benjaminmoll.com/HACT/` and website with codes `https://benjaminmoll.com/codes/`

- Continuous-time analogue of Buera and Shin (2013) `https://benjaminmoll.com/entrepreneurs_numerical/` with code `https://benjaminmoll.com/wp-content/uploads/2020/06/entrepreneurs.m`

- Something I won't talk about but everyone should be aware of
  - "Missing intercept problem" when going from cross-section to aggregates: `http://benjaminmoll.com/missing_intercept/`

Monday TA Session with Kotia

- One-hour TA session on Monday April 12th, 4pm UK / 11am ET
- Get your hands dirty and go through the codes
- Kotia will also answer any questions you may have

Resources for discrete-time HA models

1. My 1st-year PhD lecture notes
   - `https://benjaminmoll.com/Lecture2_EC442_Moll/`
   - `https://benjaminmoll.com/Lecture3_EC442_Moll/`
2. Matlab, Python & Julia codes: `http://benjaminmoll.com/HA_codes/`
   (Note: .zip file, my Google Chrome tries to block download)
   - written by Greg Kaplan in Matlab
   - translated to Python & Julia by Tom Sweeney
3. `https://quantecon.org/`, particularly Aiyagari model codes
   Python: `https://python.quantecon.org/aiyagari.html`
   Julia: `https://julia.quantecon.org/multi_agent_models/aiyagari.html`

Why Continuous Time?

Computational Advantages relative to Discrete Time

1. Borrowing constraints only show up in boundary conditions
   - FOCs always hold with "$=$"
2. "Tomorrow is today"
   - FOCs are "static", compute by hand: $c^{-\gamma} = v_a(a, y)$
3. Sparsity
   - solving Bellman, distribution = inverting matrix
   - but matrices very sparse ("tridiagonal")
   - reason: continuous time $\Rightarrow$ one step left or one step right
4. Two birds with one stone
   - tight link between solving (HJB) and (KF) for distribution
   - matrix in discrete (KF) is transpose of matrix in discrete (HJB)
   - reason: diff. operator in (KF) is adjoint of operator in (HJB)

Real Payoff: extends to more general setups

- non-convexities
- stopping time problems
- multiple assets
- transition dynamics
- aggregate shocks

Hamilton-Jacobi-Bellman Equations (Continuous-time Bellman Equations)

Reminder: Discrete-Time Bellman Equation (skip)

- Pretty much all deterministic optimal control problems in discrete time can be written as

$$V(\hat{x}_0) = \max_{\{\alpha_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, \alpha_t)$$

subject to the law of motion for the state

$$x_{t+1} = g(x_t, \alpha_t) \text{ and } \alpha_t \in A, \quad x_0 = \hat{x}_0.$$

- $\beta \in (0, 1)$: discount factor
- $x \in X \subseteq \mathbb{R}^m$: state vector
- $\alpha \in A \subseteq \mathbb{R}^k$: control vector ($\alpha$ for "action")
- $r : X \times A \to \mathbb{R}$: instantaneous return function

Reminder: Discrete-Time Bellman Equation (skip)

- Claim: the value function $V(\hat{x}_0)$ satisfies the Bellman equation

$$V(x) = \max_{\alpha} \left\{ r(x,\alpha) + \beta V(x') \quad \text{s.t.} \quad x' = g(x,\alpha) \right\}$$

- Notation: x' denotes tomorrow's state
- Important: calendar time has disappeared – "recursive notation"
- Proof sketch: consider value of optimal strategy $\{\alpha_t^*\}_{t=0}^{\infty}$

$$\begin{aligned} V(x_0) &= \sum_{t=0}^{\infty} \beta^t r(x_t, \alpha_t^*) \\ &= r(x_0, \alpha_0^*) + \sum_{t=1}^{\infty} \beta^t r(x_t, \alpha_t^*) \\ &= r(x_0, \alpha_0^*) + \beta \sum_{t=0}^{\infty} \beta^t r\left(x_{t+1}, \alpha_{t+1}^*\right) \\ &= r(x_0, \alpha_0^*) + \beta V(x_1) \end{aligned}$$

Hamilton-Jacobi-Bellman Equations

- Pretty much all deterministic optimal control problems in continuous time can be written as

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

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 ($\alpha$ for "action")
- $r : X \times A \to \mathbb{R}$: 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

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

- Compare to discrete time

$$v(k) = \max_c \; u(c) + \beta v(k') \quad k' = F(k) + (1-\delta)k - c$$

and FOC

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

Derivation from Discrete-time Bellman (skip)

- 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 (skip)

- 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 $\Delta \to 0$

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

Poisson Uncertainty

- Easy to extend this to stochastic case. Simplest case: two-state Poisson process
- Example: RBC Model. Production is $Z_t F(k_t)$ where $Z_t \in \{Z_1, Z_2\}$ Poisson with intensities λ_1, λ_2
- Result: HJB equation is

$$\rho v_i(k) = \max_c \; u(c) + v_i'(k)[Z_i F(k) - \delta k - c] + \lambda_i [v_j(k) - v_i(k)]$$

for $i = 1, 2, j \neq i$.

- Derivation similar as before. FOC

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

- Compare to discrete time

$$u'(c) = \beta \sum_{j=1}^{2} p_{ij} v_j'(k')$$

Some general, somewhat philosophical thoughts (skip)

- 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) (skip)

Recall Hamilton-Jacobi-Bellman equation:

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

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" `https://benjaminmoll.com/viscosity_for_dummies/`
- 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)

Textbook Heterogeneous-Agent Model

Textbook Heterogeneous-Agent Model

Households are heterogeneous in their wealth a and income y, solve

$$\max_{\{c_t\}_{t\geq 0}} \mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt \qquad \text{s.t.}$$

$$\dot{a}_t = y_t + r a_t - c_t$$

$$y_t \in \{y_1, y_2\} \text{ Poisson with intensities } \lambda_1, \lambda_2$$

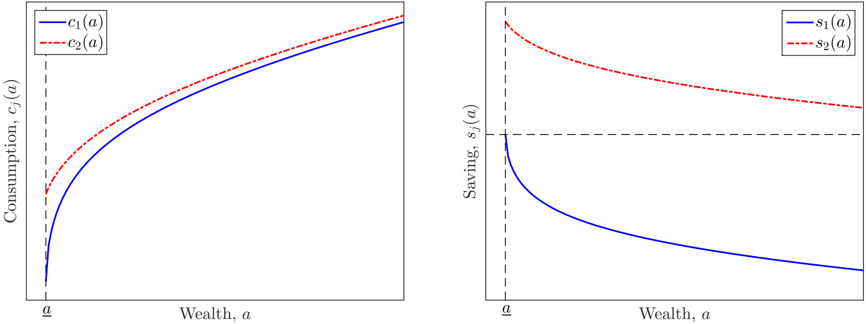
$$a_t \geq \underline{a}$$

- c_t: consumption
- u: utility function, $u' > 0$, $u'' < 0$
- ρ: discount rate
- r : interest rate
- $\underline{a} \geq -y_1/r$ if $r > 0$: borrowing limit e.g. if $\underline{a} = 0$, can only save

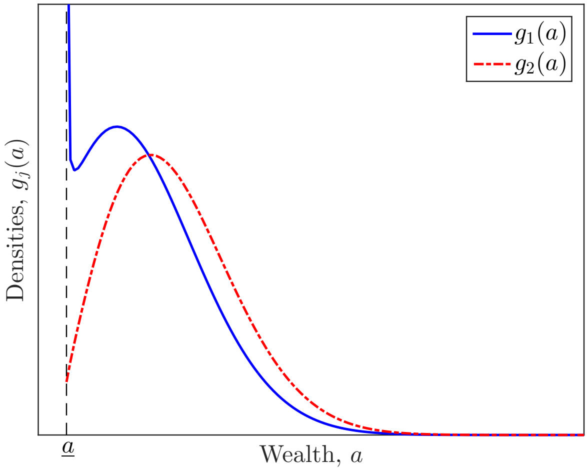
Carries over to y_t = more general processes, e.g. diffusion

Equilibrium (Huggett): bonds in fixed supply, i.e. aggregate a_t = fixed

Typical Consumption and Saving Policy Functions



Typical Stationary Distribution



Equations for Stationary Equilibrium

$$\rho v_j(a) = \max_c \; u(c) + v_j'(a)(y_j + ra - c) + \lambda_j(v_{-j}(a) - v_j(a)) \qquad \text{(HJB)}$$

$$0 = -\frac{d}{da}[s_j(a)g_j(a)] - \lambda_j g_j(a) + \lambda_{-j} g_{-j}(a), \qquad \text{(KF)}$$

$$s_j(a) = y_j + ra - c_j(a) = \text{saving policy function from (HJB)},$$

$$\int_{\underline{a}}^{\infty} (g_1(a) + g_2(a))da = 1, \quad g_1, g_2 \geq 0$$

$$S(r) := \int_{\underline{a}}^{\infty} a g_1(a) da + \int_{\underline{a}}^{\infty} a g_2(a) da = B, \qquad B \geq 0 \qquad \text{(EQ)}$$

- The two PDEs (HJB) and (KF) together with (EQ) fully characterize stationary equilibrium

Numerical Solution of HJB Equations

Codes: `https://benjaminmoll.com/codes/`

One-Slide Summary of Numerical Method

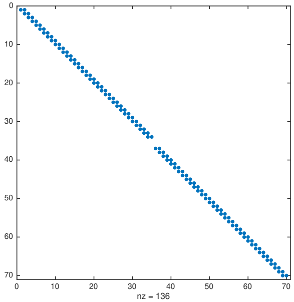
- Consider general HJB equation:

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

- 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 (skip)

- 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
  `http://benjaminmoll.com/barles-souganidis/`
- 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

    `https://benjaminmoll.com/HJB_NGM/`

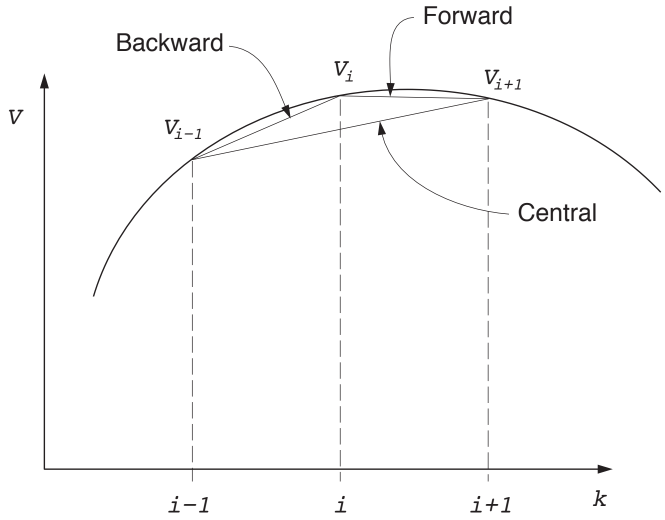
    `https://benjaminmoll.com/HJB_NGM_implicit/`

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:

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

Finite Difference Approximations to $v'(k_i)$



Note: we'll use only backward and forward, central never used

Finite Difference Approximation

FD approximation to HJB is

$$\rho v_i = u(c_i) + v_i' s_i, \quad s_i := F(k_i) - \delta k_i - c_i, \quad c_i = (u')^{-1}(v_i') \qquad (*)$$

for $i = 1, ..., I$ and where

- s_i denotes saving at grid point i
- v_i' is either backward or forward FD approximation

Questions:

- Which FD approximation – backward or forward – should we use?
- ... and where in the state space?

Turns out this is extremely important. Good solution $\Rightarrow$ next slide

- technical reason: Barles-Souganidis monotonicity condition

Upwinding

- Which FD approximation you use is extremely important
- Best solution: use so-called "upwind scheme." Basic idea:
  - forward difference whenever drift of state variable positive
  - backward difference whenever drift of state variable negative

- Upwind version of $(*)$ from previous slide

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

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

- This ignores two complications
  1. $(**)$ has circular element: saving s_i itself depends on forward or backward approx ($s_i = F(k_i) - \delta k_i - c_i$ and $c_i = (u')^{-1}(v_i')$)
  2. $(**)$ is extremely non-linear $\Rightarrow$ need to solve iteratively

Put these complications aside for now – revisit in a few slides

The matrix **A**

- Recall

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

- Can write this in matrix notation

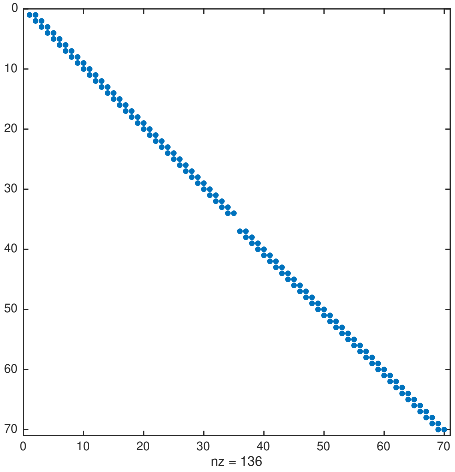
$$\rho v_i = u(c_i) + \begin{bmatrix} -\frac{s_i^-}{\Delta k} & \frac{s_i^-}{\Delta k} - \frac{s_i^+}{\Delta k} & \frac{s_i^+}{\Delta k} \end{bmatrix} \begin{bmatrix} v_{i-1} \\ v_i \\ v_{i+1} \end{bmatrix}$$

and hence

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

where **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:

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

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

Revisiting the Two Complications

- Recall discretized HJB equation in matrix form:

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

- If this were whole story, could immediately solve $(\rho \mathbf{I} - \mathbf{A})\,\mathbf{v} = \mathbf{u}$
- But it isn't whole story because c_i and s_i depend on v_i'
$\Rightarrow$ really $\mathbf{A}$ (and $\mathbf{u}$) depend on $\mathbf{v}$

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

- Two complications
  1. circular element: how construct $\mathbf{A}$ given saving s_i itself depends on forward or backward approximation?
  2. extremely non-linear $\Rightarrow$ how to solve iteratively?

1. Construction of $\mathbf{A}$ given c_i, s_i depends on v_i' (skip)

- Use short-hand notation: $v'_{i,B} := \frac{v_i - v_{i-1}}{\Delta k}$ and $v'_{i,F} := \frac{v_{i+1} - v_i}{\Delta k}$
- Key idea: c_i, s_i should be consistent with upwind scheme. Define:

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

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

$$c_i = c_{i,F}\mathbf{1}_{\{s_{i,F}>0\}} + c_{i,B}\mathbf{1}_{\{s_{i,B}<0\}} + \bar{c}_i\mathbf{1}_{\{s_{i,F}<0<s_{i,B}\}}$$

  where $\mathbf{1}_{\{\cdot\}}$ is indicator function, and $\bar{c}_i = F(k_i) - \delta k_i$
- Where does $\bar{c}_i = F(k_i) - \delta k_i$ 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 $s_i = 0 \Rightarrow c_i = F(k_i) - \delta k_i$ (steady state)
- Upwind finite difference approximation is

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

- $\Rightarrow$ Entries of $\mathbf{A}(\mathbf{v})$ are $-\frac{s_{i,B}^-}{\Delta k}$, $\frac{s_{i,B}^-}{\Delta k} - \frac{s_{i,F}^+}{\Delta k}$, and $\frac{s_{i,F}^+}{\Delta k}$

2. Iterative solution to $\rho \mathbf{v} = \mathbf{u}(\mathbf{v}) + \mathbf{A}(\mathbf{v})\mathbf{v}$ (skip)

Two ways of iterating:

1. Explicit method: slightly easier to explain/implement but inefficient

2. Implicit method: much more efficient

Always choose the implicit method!!

2. Explicit method `https://benjaminmoll.com/HJB_NGM/` (skip)

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

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

- Algorithm: Guess v_i^0, $i=1,...,I$ and for $n=0,1,2,...$ follow
  1. Compute $c_i^n, s_{i,F}^n, s_{i,B}^n$ as I just explained
  2. Find $\mathbf{v}^{n+1}$ from $(*)$
  3. If $\mathbf{v}^{n+1}$ is close enough to $\mathbf{v}^n$: stop. Otherwise, go to step 1.

- In matrix form

$$\frac{\mathbf{v}^{n+1}-\mathbf{v}^n}{\Delta}+\rho\mathbf{v}^n = \mathbf{u}(\mathbf{v}^n)+\mathbf{A}(\mathbf{v}^n)\mathbf{v}^n$$

- Important parameter: Δ = step size, cannot be too large ("CFL condition")
- Pretty inefficient: I need 5,990 iterations (though quite fast)

2. Implicit Method `https://benjaminmoll.com/HJB_NGM_implicit/` (skip)

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

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

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

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta} + \rho \mathbf{v}^{n+1} = \mathbf{u}(\mathbf{v}^n) + \mathbf{A}(\mathbf{v}^n)\mathbf{v}^{n+1}$$

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

- but $\mathbf{A}(\mathbf{v}^n)$ is super sparse $\Rightarrow$ super fast

- 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 (skip)

- 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

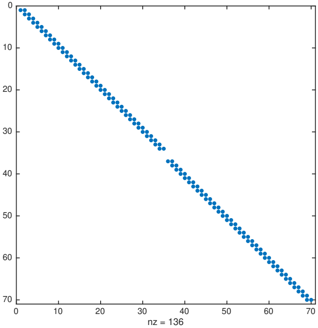
- Consider general HJB equation:

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

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

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

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



Computations for Heterogeneous Agent Model

Computations for Heterogeneous Agent Model

- Hard part: HJB equation
- Easy part: KF equation. Once you solved HJB equation, get KF equation "for free"
- System to be solved

$$\rho v_j(a) = \max_c \; u(c) + v_j'(a)(y_j + ra - c) + \lambda_j(v_{-j}(a) - v_j(a)), \quad j = 1, 2$$

$$0 = -\frac{d}{da}[s_j(a)g_j(a)] - \lambda_j g_j(a) + \lambda_{-j} g_{-j}(a), \quad j = 1, 2$$

$$B = \int_{\underline{a}}^{\infty} a g_1(a) da + \int_{\underline{a}}^{\infty} a g_2(a) da := S(r)$$

Summary: Algorithm for Stationary Equilibria

- Use finite difference method: `https://benjaminmoll.com/codes/`
- Discretize state space a_i, $i = 1, ..., I$ with step size Δa

$$v_j'(a_i) \approx \frac{v_{i+1,j} - v_{i,j}}{\Delta a} \quad \text{or} \quad \frac{v_{i,j} - v_{i-1,j}}{\Delta a}$$

$$\text{Denote} \quad \mathbf{v} = \begin{bmatrix} v_1(a_1) \\ \vdots \\ v_2(a_I) \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} g_1(a_1) \\ \vdots \\ g_2(a_I) \end{bmatrix}, \quad \text{dimension} = 2I \times 1$$

- End product of FD method: system of sparse matrix equations

$$\rho \mathbf{v} = \mathbf{u}(\mathbf{v}) + \mathbf{A}(\mathbf{v}; r)\mathbf{v}$$
$$\mathbf{0} = \mathbf{A}(\mathbf{v}; r)^{\mathrm{T}} \mathbf{g}$$
$$B = S(\mathbf{g}; r)$$

which is easy to solve on computer

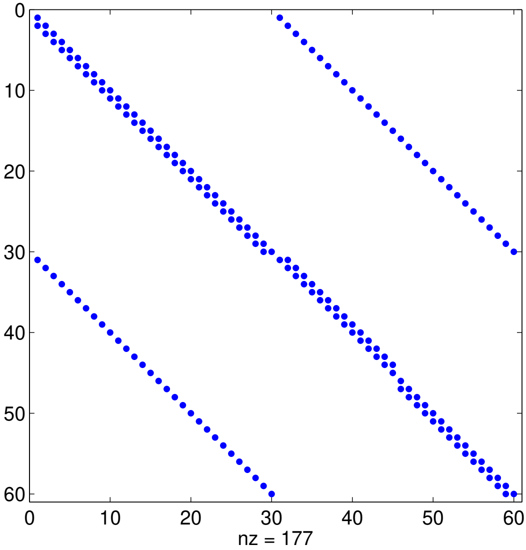
Computing the HJB Equation

- As before, discretized HJB equation is

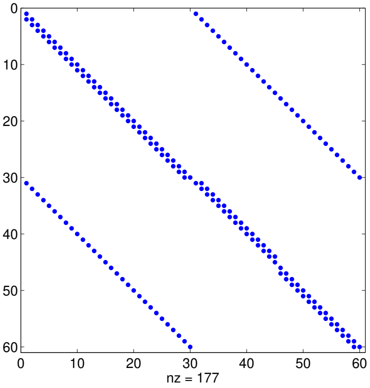
$$\rho \mathbf{v} = \mathbf{u}(\mathbf{v}) + \mathbf{A}(\mathbf{v})\mathbf{v} \tag{HJBd}$$

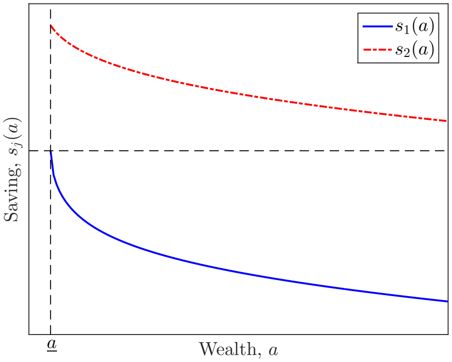
- $\mathbf{A}$ is $N \times N$ transition matrix
  - here $N = 2 \times I$, I=number of wealth grid points
  - $\mathbf{A}$ depends on $\mathbf{v}$ (nonlinear problem)
  - solve using implicit scheme

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



Exercise: explain structure of **A** from saving policy fn





Computing the KF Equation

- Equations to be solved

$$0 = -\frac{d}{da}[s_1(a)g_1(a)] - \lambda_1 g_1(a) + \lambda_2 g_2(a)$$
$$0 = -\frac{d}{da}[s_2(a)g_2(a)] - \lambda_2 g_2(a) + \lambda_1 g_1(a)$$

  with $1 = \int_{\underline{a}}^{\infty} g_1(a)da + \int_{\underline{a}}^{\infty} g_2(a)da$

- Actually, super easy: discretized version is simply

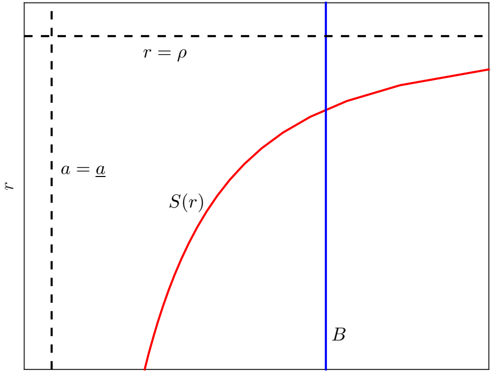
$$0 = \mathbf{A}(\mathbf{v})^{\mathrm{T}}\mathbf{g} \qquad \text{(KFd)}$$

  - eigenvalue problem
  - get KF for free, one more reason for using implicit scheme
- Why transpose?
  - operator in (HJB) is “adjoint” of operator in (KF)
  - “adjoint” = infinite-dimensional analogue of matrix transpose
- In principle, can use similar strategy in discrete time

Finding the Equilibrium Interest Rate

Use any root-finding method, e.g. bisection method

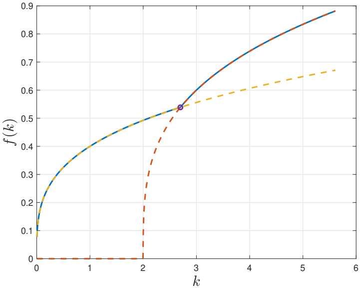
- increase r whenever $S(r) < B$
- decrease r whenever $S(r) > B$



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



- See section 5.2 of Banerjee and Duflo (2005) for similar model

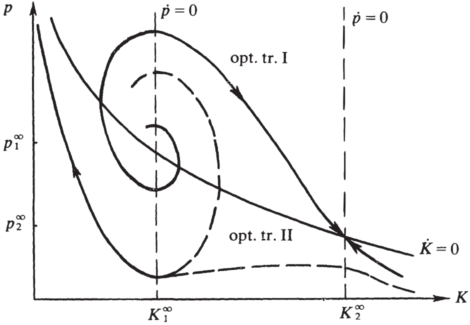
Standard Methods

- Discrete time: first-order conditions

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

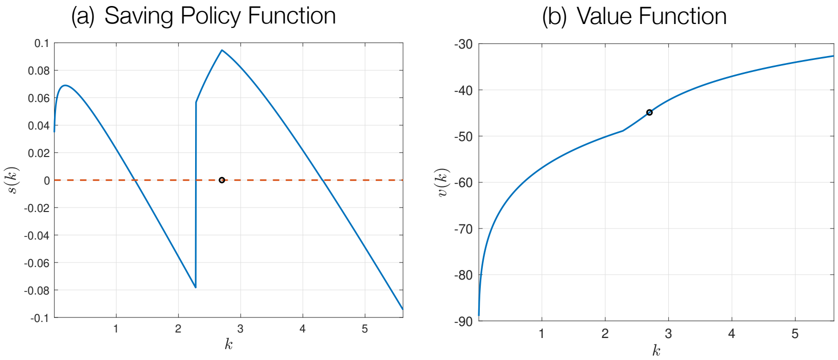
  no longer sufficient, typically multiple solutions

- Continuous time: Skiba (1978)

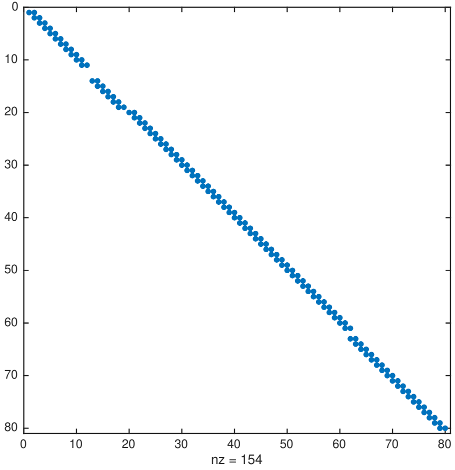


Instead: Using Finite-Difference Scheme

Nothing changes, use same exact algorithm as for growth model with concave production function `https://benjaminmoll.com/HJB_NGM_skiba/`

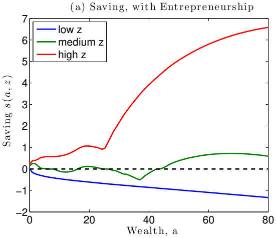


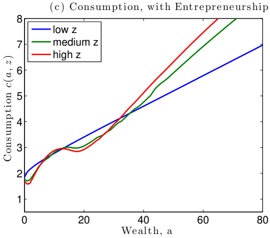
Visualization of **A** (output of `spy(A)` in Matlab)

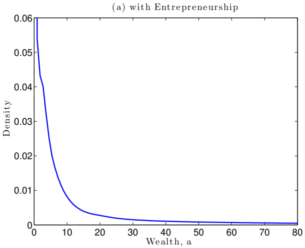


Occupational Choice

1. Model on my website (cont-time version of Buera & Shin, 2013)
`https://benjaminmoll.com/entrepreneurs_numerical/` with code
`https://benjaminmoll.com/wp-content/uploads/2020/06/entrepreneurs.m`







2. Cavalcanti, Kaboski, Martins and Santos (2021)

- Tiago, Joe, Bruno & Cezar were kind enough to share code so you can play around with it yourself! `http://benjaminmoll.com/ckms_code/`
(Note: .zip file, my Google Chrome tries to block download)