Heterogeneous Agent Models in Continuous Time
Part I

Benjamin Moll
Princeton

Rochester, 1 March 2017

What this lecture is about

- Many interesting questions require thinking about distributions
  - Why are income and wealth so unequally distributed?
  - Is there a trade-off between inequality and economic growth?
  - What are the forces that lead to the concentration of economic activity in a few very large firms?
- Modeling distributions is hard
  - closed-form solutions are rare
  - computations are challenging
- Goal: teach you some new methods that make progress on this
  - solving heterogeneous agent model = solving PDEs
  - main difference to existing continuos-time literature: handle models for which closed-form solutions do not exist
  - based on joint work with Yves Achdou, SeHyoun Ahn, Jiequn Han, Greg Kaplan, Pierre-Louis Lions, Jean-Michel Lasry, Gianluca Violante, Tom Winberry, Christian

Solving het. agent model = solving PDEs

- More precisely: a system of two PDEs
  1. Hamilton-Jacobi-Bellman equation for individual choices
  2. Kolmogorov Forward equation for evolution of distribution

- Many well-developed methods for analyzing and solving these
  - codes: `http://www.princeton.edu/~moll/HACTproject.htm`

- Apparatus is very general: applies to any heterogeneous agent model with continuum of atomistic agents
  1. heterogeneous households (Aiyagari, Bewley, Huggett,...)
  2. heterogeneous producers (Hopenhayn,...)

- can be extended to handle aggregate shocks (Krusell-Smith,...)

Outline

Lecture 1

1. Refresher: HJB equations
2. Textbook heterogeneous agent model
3. Numerical solution of HJB equations
4. Models with non-convexities (Skiba)

Lecture 2

1. Analysis and numerical solution of heterogeneous agent model
2. Transition dynamics/MIT shocks
3. Stopping time problems
4. Models with multiple assets (HANK)

"When Inequality Matters for Macro and Macro Matters for Inequality"

1. Aggregate shocks via perturbation (Reiter)
2. Application to consumption dynamics

Computational Advantages relative to Discrete Time

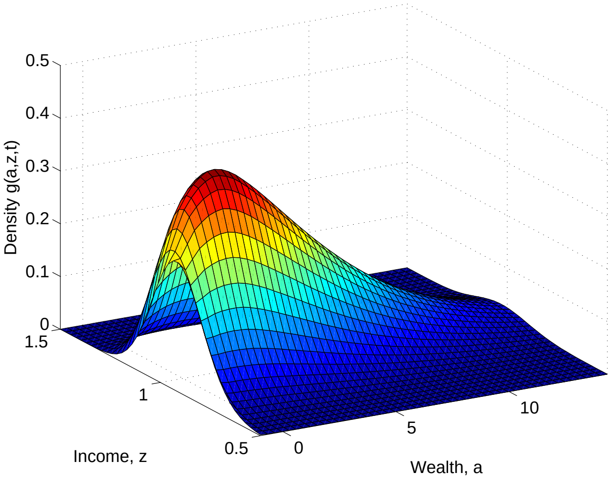
1. Borrowing constraints only show up in boundary conditions
    - FOCs always hold with "$=$"
2. "Tomorrow is today"
    - FOCs are "static", compute by hand: $c^{-\gamma} = v_a(a, y)$
3. Sparsity
    - solving Bellman, distribution = inverting matrix
    - but matrices very sparse ("tridiagonal")
    - 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 (no need for threshold rules)
- multiple assets
- aggregate shocks

What you'll be able to do at end of this lecture

- Joint distribution of income and wealth in Aiyagari model



What you'll be able to do at end of this lecture

- Experiment: effect of one-time redistribution of wealth

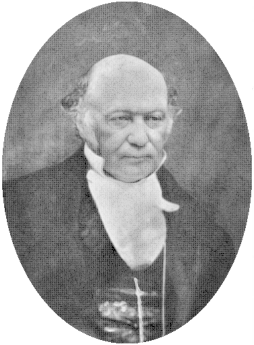
What you'll be able to do at end of this lecture

Video of convergence back to steady state

https://www.dropbox.com/s/op5u2nlifmmer2o/distribution_tax.mp4?dl=0

Review: HJB Equations

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

- 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
- $x \in X \subseteq \mathbb{R}^m$: state vector
- $\alpha \in A \subseteq \mathbb{R}^n$: control vector
- $r : X \times A \to \mathbb{R}$: instantaneous return function

Example: Neoclassical Growth Model

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

subject to

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

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 $m > 1$, $v'(x) \in \mathbb{R}^m$ is the gradient vector 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 equation

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

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 you 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

A Textbook Heterogeneous Agent Model

Households

are heterogeneous in their wealth a and income y, solve

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

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

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

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

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

later: carries over to y_t = general diffusion process.

Equations for Stationary Equilibrium

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

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

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

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

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

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

Transition Dynamics

- Needed whenever initial condition $\neq$ stationary distribution
- Equilibrium still coupled systems of HJB and KF equations...
- ... but now time-dependent: $v_j(a, t)$ and $g_j(a, t)$
- See paper for equations
- Difficulty: the two PDEs run in opposite directions in time
  - HJB looks forward, runs backwards from terminal condition
  - KF looks backward, runs forward from initial condition

Numerical Solution of HJB Equations

Finite Difference Methods

- See `http://www.princeton.edu/~moll/HACTproject.htm`
- Explain using neoclassical growth model, easily generalized to heterogeneous agent models

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

- 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`

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”
- Background on viscosity soln’s: “Viscosity Solutions for Dummies”
  `http://www.princeton.edu/~moll/viscosity_slides.pdf`

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

$$v_i = v(k_i)$$

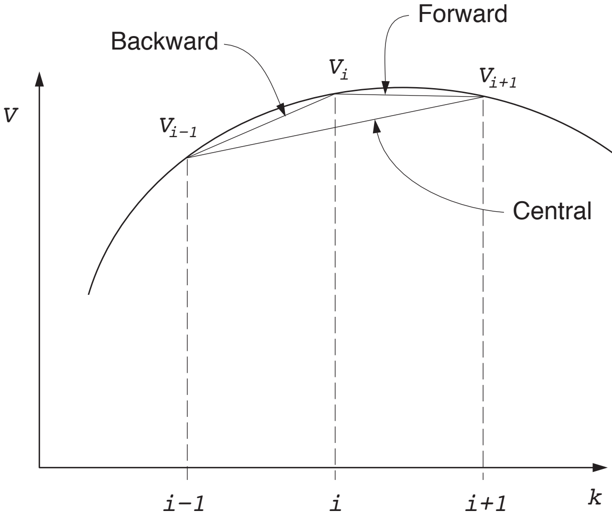
- Need to approximate $v'(k_i)$.
- Three different possibilities:

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

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

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

Finite Difference Approximations to $v'(k_i)$



Finite Difference Approximation

FD approximation to HJB is

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

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
- slides on my website: 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

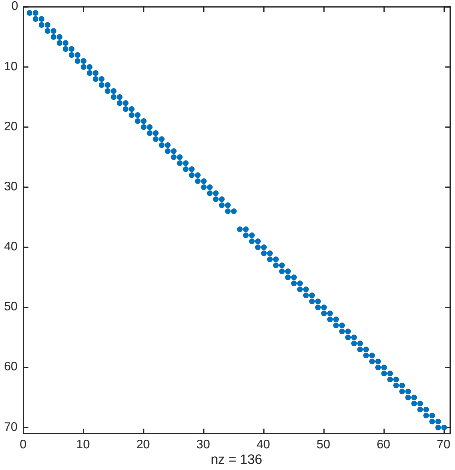
$$\rho v_i = u(c_i) + \begin{bmatrix} -\frac{s_{i,B}^-}{\Delta k} & \frac{s_{i,B}^-}{\Delta k} - \frac{s_{i,F}^+}{\Delta k} & \frac{s_{i,F}^+}{\Delta k} \end{bmatrix} \begin{bmatrix} v_{i-1} \\ v_i \\ v_{i+1} \end{bmatrix}$$

and hence

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

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

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



The matrix $\mathbf{A}$

- FD method approximates process for k with discrete Poisson process, $\mathbf{A}$ 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

- $\mathbf{A}$ (and $\mathbf{u}$) depend on $\mathbf{v}$ (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 v^n, update 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 v^{n+1} from $(*)$.
  4. If v^{n+1} is close enough to 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

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

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

- but $\mathbf{A}(\mathbf{v}^{\mathrm{n}})$ 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);
```

Relation to Kushner-Dupuis "Markov-Chain Approx"

- There's another common method for solving HJB equation: "Markov Chain Approximation Method"
  - Kushner and Dupuis (2001) "Numerical Methods for Stochastic Control Problems in Continuous Time"
  - effectively: convert to discrete time, use value fn iteration
- FD method not so different: also converts things to "Markov Chain"

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

- Connection between FD and MCAM
  - see Bonnans and Zidani (2003), "Consistency of Generalized Finite Difference Schemes for the Stochastic HJB Equation"
  - also shows how to exploit insights from MCAM to find FD scheme satisfying Barles-Souganidis conditions
- Another source of useful notes/codes: Frédéric Bonnans' website

`http://www.cmap.polytechnique.fr/~bonnans/notes/edpfin/edpfin.html`

Non-Convexities

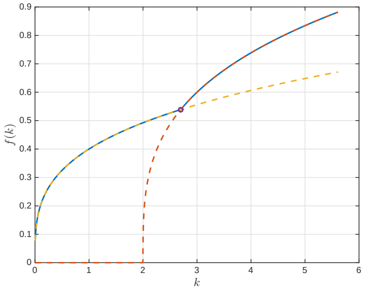
Non-Convexities

- 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"

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

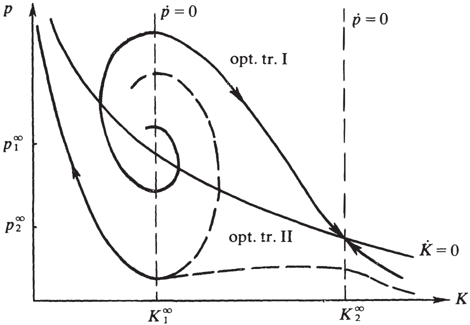


Standard Methods

- Discrete time: first-order conditions

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

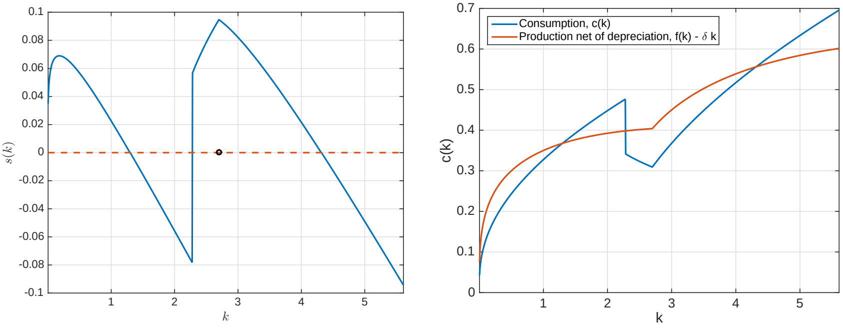
  no longer sufficient, typically multiple solutions
  - some applications: sidestep with lotteries (Prescott-Townsend)
- Continuous time: Skiba (1978)



Instead: Using Finite-Difference Scheme

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

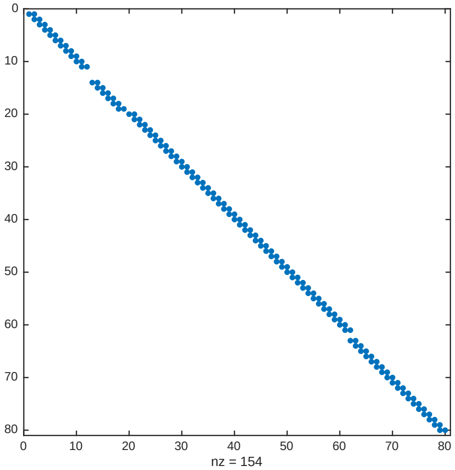
`http://www.princeton.edu/~moll/HACTproject/HJB_NGM_skiba.m`



(a) Saving Policy Function

(b) Consumption Policy Function

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



Appendix

Derivation from Discrete-time Bellman ‣ Back

- 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 $\Delta \to 0$), $e^{-\rho\Delta} \approx 1 - \rho\Delta$

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

- 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 $\Delta \to 0$

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

Derivation of Poisson KF Equation ▸ Back

- Work with CDF (in wealth dimension)

$$G_j(a,t) := \Pr(\tilde{a}_t \le a, \tilde{y}_t = y_j)$$

- Income switches from y_j to y_{-j} with probability $\Delta\lambda_j$
- Over period of length Δ, wealth evolves as $\tilde{a}_{t+\Delta} = \tilde{a}_t + \Delta s_j(\tilde{a}_t)$
- Similarly, answer to question "where did $\tilde{a}_{t+\Delta}$ come from?" is

$$\tilde{a}_t = \tilde{a}_{t+\Delta} - \Delta s_j(\tilde{a}_{t+\Delta})$$

- Momentarily ignoring income switches and assuming $s_j(a) < 0$

$$\Pr(\tilde{a}_{t+\Delta} \le a) = \underbrace{\Pr(\tilde{a}_t \le a)}_{\text{already below } a} + \underbrace{\Pr(a \le \tilde{a}_t \le a - \Delta s_j(a))}_{\text{cross threshold } a} = \Pr(\tilde{a}_t \le a - \Delta s_j(a))$$

- Fraction of people with wealth below a evolves as

$$\begin{aligned}\Pr(\tilde{a}_{t+\Delta} \le a, \tilde{y}_{t+\Delta} = y_j) = (1-\Delta\lambda_j)&\Pr(\tilde{a}_t \le a - \Delta s_j(a), \tilde{y}_t = y_j)\\ +\Delta\lambda_{-j}&\Pr(\tilde{a}_t \le a - \Delta s_{-j}(a), \tilde{y}_t = y_{-j})\end{aligned}$$

- Intuition: if have wealth $< a - \Delta s_j(a)$ at t, have wealth $< a$ at $t + \Delta$

Derivation of Poisson KF Equation

- Subtracting $G_j(a,t)$ from both sides and dividing by Δ

$$\frac{G_j(a,t+\Delta)-G_j(a,t)}{\Delta} = \frac{G_j(a-\Delta s_j(a),t)-G_j(a,t)}{\Delta} - \lambda_j G_j(a-\Delta s_j(a),t) + \lambda_{-j} G_{-j}(a-\Delta s_{-j}(a),t$$

- Taking the limit as $\Delta \to 0$

$$\partial_t G_j(a,t) = -s_j(a)\partial_a G_j(a,t) - \lambda_j G_j(a,t) + \lambda_{-j} G_{-j}(a,t)$$

where we have used that

$$\lim_{\Delta\to 0} \frac{G_j(a-\Delta s_j(a),t)-G_j(a,t)}{\Delta} = \lim_{x\to 0} \frac{G_j(a-x,t)-G_j(a,t)}{x} s_j(a)$$
$$= -s_j(a)\partial_a G_j(a,t)$$

- Intuition: if $s_j(a) < 0$, $\Pr(\tilde{a}_t \leq a, \tilde{y}_t = y_j)$ increases at rate $g_j(a,t)$
- Differentiate w.r.t. a and use $g_j(a,t) = \partial_a G_j(a,t) \Rightarrow$

$$\partial_t g_j(a,t) = -\partial_a[s_j(a,t)g_j(a,t)] - \lambda_j g_j(a,t) + \lambda_{-j} g_{-j}(a,t)$$