Continuous Time

Juan Herreño Johannes Wieland

UCSD, Spring 2024

Outline

Outline

Caveats

- Lecture designed to take you to the computer as fast as possible
- Not a lecture on continuous time calculus
- Instead what Ben Moll calls a *cookbook approach*
- I borrowed heavily from Ben's material
- If you are interested in the math behind these problems the reference is Achdou et al (2022) in Restud.

Two new friends

- In general, when you write macro models in continuous time you need two equations (PDEs)
  1. Hamilton-Jacobi-Bellman equation: individual choices
  2. Kolmogorov Forward equation: How the distribution of agents evolves
- Computational differences of CT versus DT
  1. *Today is tomorrow:* FOCs are static. Good thing, you can use policy functions without computing the expectation of the realization of tomorrow, which involve integrals, laws of motion, interpolations. All expensive!
  2. In one instant you cannot move too much: transitions in the **state-space** are very sparse. Computers love sparse matrices.
  3. Solution of the HJB and KF are tightly linked. Compute one, get the other for free.

From Bellman to HJB

State x. Length of period t is 1 unit of time

- Start with the DT Bellman equation

$$v(x_t) = \max_{y_t} u(y_t) + \beta v(x_{t+1}) \text{ for } y_t \in \Gamma(x, x_{t+1})$$

- Allow for periods of time to be different than 1. Call them Δ

$$v(x_t) = \max_{y_t} \Delta u(y_t) + \beta(\Delta) v(x_{t+\Delta}) \text{ for } y_t \in \Gamma(x_t, x_{t+\Delta})$$

- Make clear what the function $\beta(\Delta)$ is:

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

- with useful properties

$$\lim_{\Delta \to 0} \beta(\Delta) = 1, \lim_{\Delta \to \infty} \beta(\Delta) = 0, \beta(1) = \beta$$

From Bellman to HJB

- for small Δ, $e^{-\rho\Delta} \approx 1 - \rho\Delta$
- From the Bellman equation subtract $(1-\rho\Delta)v(x_t)$

$$\rho\Delta v(x_t) = \max_{y_t} \Delta u(y_t) + (1-\rho\Delta)(v(x_{t+\Delta}) - v(x_t))$$

- Divide by Δ and divide and multiply last term by $x_{t+\Delta} - x_t$

$$\rho v(x_t) = \max_{y_t} u(y_t) + (1-\rho\Delta)\frac{v(x_{t+\Delta}) - v(x_t)}{x_{t+\Delta} - x_t}\frac{x_{t+\Delta} - x_t}{\Delta}$$

- Take Δ to zero

$$\rho v(x_t) = \max_{y_t} u(y_t) + v'(x_t)\dot{x}_t$$

This is the HJB. We will hold off on the KF equation, not thinking about distributions yet

Example with the Neoclassical growth model

In our previous notation

1. k takes the place of x
2. $\dot{k} = i - \delta k = F(k) - c - \delta k$
3. So the HJB equation is

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

4. With a first order condition

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

5. No k' anywhere
6. Basics of the problem. If you:
    - give me v
    - and a way to calculate v'
    - then I can tell you c

Outline

Challenges

We are still working with computers, so we must discretize v

- and add the challenge of computing v'!

Imagine that we have a guess for v in a grid of points $k_1, k_2, ..., k_n$.

- We can compute approximation of the derivative using "finite differences". Three possibilities
  1. $v'(k_i) \approx \frac{v_i - v_{i-1}}{k_i - k_{i-1}} = v'_{i,B}$. B for *backward difference*
  2. $v'(k_i) \approx \frac{v_{i+1} - v_i}{k_{i+1} - k_i} = v'_{i,F}$. F for *forward difference*
  3. $v'(k_i) \approx \frac{v_{i+1} - v_{i-1}}{k_{i+1} - k_{i-1}} = v'_{i,C}$. C for *central difference*
- For an equally space grid $k_{i+1} - k_i = k_i - k_{i-1} = \Delta k$
  1. $v'(k_i) \approx \frac{v_i - v_{i-1}}{\Delta k} = v'_{i,B}$. B for *backward difference*
  2. $v'(k_i) \approx \frac{v_{i+1} - v_i}{\Delta k} = v'_{i,F}$. F for *forward difference*
  3. $v'(k_i) \approx \frac{v_{i+1} - v_{i-1}}{2\Delta k} = v'_{i,C}$. C for *central difference*

SUMMARY

- For a given point in the grid i

$$\rho v(k_i) = u(c_i) + v_i'(F(k) - \delta k - c_i)$$

- There is no max operator because we will use the FOC

$$c_i = (u')^{-1} v_i'$$

- We have two things to resolve

  1. The HJB holds for the right v, but we do not know v. Need to spell out the algorithm to iterate
  2. Which of our three candidates v' should we use?

The Upwind Scheme

- Rule of thumb: forward difference when the state drifts in the positive direction. backward difference when the state drifts in the negative direction
- Where the drift s_i is nothing more than the law of motion of the state

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

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

- First try: approximate v' using the upwind scheme

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

- Issue, what happens when $s_{i,F} < 0$ and $s_{i,B} > 0$
- for concave functions (like v), $s_{i,F} < s_{i,B}$, so we are going to interpret this case as the drift being zero, we are at the steady state

The Upwind Scheme

So instead of using

$$v_i' = v_{i,F}' \mathbb{1}_{s_{i,F}>0} + v_{i,B}' \mathbb{1}_{s_{i,B}<0}$$

we will use:

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

where $\bar{v}_i'$ is given by u' at the point where $\dot{k} = 0$

$$\bar{v}_i' = u'(F(k) - \delta k)$$

Turns out that the upwind scheme is extremely important. Not a matter of efficiency, it is a matter of eventually getting to the right v.
Lesson: When you are more sofisticated (use FOCs), you need to be more careful, where careful here means being consistent in your choice of v' depending on the drift of the state x.

Exercise for the lab

$$\max \int_0^\infty e^{-\rho t} u(c(t)) dt \tag{1}$$

$$u(c) = \log(c) \tag{2}$$

$$\dot{a}(t) = ra(t) + y(t) - c(t) \tag{3}$$

$$y(t) = 1 \tag{4}$$

$$\beta = e^{-\rho} = 0.99 \tag{5}$$

$$r = \rho \tag{6}$$

Task:

- Write a function vprime_upwind that takes as an input a function v, plus the grids and parameters I provide you, and produces as an output v' using the upwind scheme

Outline

Explicit Method in three steps

Start from a guess v^n (n for the n-th time you've tried different guesses),

1. Use the upwind scheme to compute v_i'
2. From the upwind scheme v_i', compute $c_i = u^{-1}(v_i')$
3. Compute the difference between current guess ρv_i^n and the new iteration

$$d_i^n = u(c_i^n) + (v_i^n)'(k_i)(F(k_i) - \delta k_i - c_i) - \rho v_i^n$$

4. Update the value function in the following way

$$v_i^{n+1} = v_i^n + \tilde{\Delta} d_i^n$$

5. Intuition: change v^n by a factor of d_i^n
6. If v^n and v^{n+1} are close, stop.

Warning: Δ must be *small*

Exercise for the lab

$$\max \int_0^\infty e^{-\rho t} u(c(t)) dt \tag{7}$$

$$u(c) = \log(c) \tag{8}$$

$$\dot{a}(t) = ra(t) + y(t) - c(t) \tag{9}$$

$$y(t) = 1 \tag{10}$$

$$\beta = e^{-\rho} = 0.99 \tag{11}$$

$$r = \rho \tag{12}$$

Details for the grid:

- 100 points for a
- between 0.1 and 1
- $\tilde{\Delta} = 0.001$

Task:

- Solve using the explicit method

Outline

Writing the HJB in matrix form

- So we can write the HJB equation as

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

- with the notation $x^{+} = \max(x, 0)$ and $x^{-} = min(x, 0)$
- Convince yourselves at home that this is equivalent to this vector expression

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

- Stack everything on a matrix

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

with $\mathbf{A}$ a square matrix with as many points you have in your grid for k

Writing the HJB in matrix form

- Tempting to think you can solve for v directly from the equation

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

- Not feasible. A is not a set of constants. Terms in $\mathbf{A}$ that depend on c, and therefore on v'.

- Notice that for a grid point i, only three entries of A are non-zero. $i-1$, i, $i+1$. $\mathbf{A}$ is a *sparse* matrix

- Intuition. In any given instant, an agent may be moving one step to the right, one step to the left, or stay in the same position, but no jumps in an instant.

Implicit method is faster

This material is for you. You will implement it in your homework.
Instead of:

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

do

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

which in matrix form is

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

$\tilde{\Delta}$ does not need to be as small. Way more efficient. Need to make explicit that A is sparse.

Outline

Simulate given a price

- After you have solved the HJB problem you can simulate it
- There are many uses for this

1. Compute distributions of state-space variables across agents (next week we will see ways to do this in a more efficient way)
2. Compute moments from you model. Useful for the "run the same regression in your model as in the data" approach
3. More in general, useful for SMM. Estimate parameters to match some moments in the data from the model.

Simulate given a price

- In our problem we obtained a policy function $c(a,\Theta)$ for Θ parameters.
- Given an a_0, we can use the policy function to get a_1

$$\dot{a}_t = ra_t + y - c_t$$

- we first need to transform it into:

$$a_{t+\delta} = a_t + \delta\left(ra_t + y - c_t\right)$$

- for a small δ

Exercise for the lab

- Use the prompt I give you to produce the following simulation:
  - 100 agents that start uniformly over the a vector
  - Interpolate the policy function c over 10,000 points in the same range of a
  - Simulate for 5 years, with a δ of 1/100 years.
  - plot the path of consumption of the 100 consumers
  - run a cross-sectional regression for each period

    $$c_i = \beta_0 + \beta_1 a_i + \xi_i$$

    plot the time series of your estimate of $\hat{\beta}_1$.
  - try when $r = \rho$, try when $r = 0$