

# OSM LAB - DYNAMIC PROGRAMMING

Jason DeBacker

June 18-22, 2018

Notes for Lecture #3

Today- Applications:

- HH problem - portfolio choice
- Firm Dynamics
- Labor search

Discrete Choice Cake Eating Problem: (an example of an optimal stopping problem)

- control: {eat cake, leave cake}  $\rightarrow$  binary (0,1 choice)
  - $z \in \{1, 0\}$
- state:  $w, \varepsilon \Rightarrow$  know  $w$  and  $\varepsilon$  at the time of the decision
- transition:  $w' = \rho w$  if  $z = 0$  (grow/shrink leftover cake),  $w' = 0$  if  $z = 1$  (cake eaten in period 1, no  $w'$ )
- value function:  $V(w, \varepsilon) = \max\{\underbrace{V^0(w, \varepsilon)}_{\text{leave cake}}, \underbrace{V^1(w, \varepsilon)}_{\text{eat cake}}\}, \forall(w, \varepsilon)$ 
  - $V^0(w, \varepsilon) = \beta E_{\varepsilon'|\varepsilon} V(\rho w, \varepsilon')$
  - $V^1(w, \varepsilon) = \varepsilon u(w)$
- policy function:  $z(w, \varepsilon) \in \{0, 1\}, \forall(w, \varepsilon)$
- Choice depends on:
  - State variables: ( $w$  &  $\varepsilon$ ) b/c in state vector
  - Parameters:
    - \*  $\rho$ , b/c as  $\rho \uparrow$ , gain to waiting
    - \*  $\beta$ ,  $\beta \downarrow$  cost to waiting
    - \*  $\Pi$ : the transition matrix
- NOTE: No Euler equation in discrete case - eat or don't eat - it's not continuous
- e.g.,  $\rho = 1, \varepsilon \in \{\varepsilon_L, \varepsilon_H\}$ 
  - $z(w, \varepsilon_H) = 1, \forall w$ : nothing to wait for!
  - $z(w, \varepsilon_L) = \{0, 1\} \rightarrow$  wait if:  $\beta$  near 1 or  $\pi_{LH}$  sufficiently high
    - \* NOTE:  $w$  unimportant b/c its in both  $V^0$  and  $V^1$  decisions
    - \* How high does  $\pi_{LH}$  have to be to wait?
    - \* wait if:  $\varepsilon_L u(w) \leq \beta \{E_{\varepsilon'|\varepsilon_L} V(w, \varepsilon')\} = \beta \{\pi_{LH} \varepsilon_H u(w) + \pi_{LL} V(w, \varepsilon_L)\}$
    - \* B/c always eat in high, and assuming never eat in low (this is the RHS of the equality), know that:  $E_{\varepsilon'|\varepsilon_L} V(w, \varepsilon') = \pi_{LH} \varepsilon_H u(w) + \pi_{LL} \beta E_{\varepsilon'|\varepsilon_L} V(w, \varepsilon')$
    - \* Solving for  $V(w, \varepsilon') \Rightarrow E_{\varepsilon'|\varepsilon_L} V(w, \varepsilon') = \frac{\pi_{LH} \varepsilon_H u(w)}{1 - \beta \pi_{LL}}$
    - \* Thus, wait if  $\varepsilon_L u(w) \leq \beta \{E_{\varepsilon'|\varepsilon_L} V(w, \varepsilon')\} = \frac{\beta \pi_{LH} \varepsilon_H u(w)}{1 - \beta \pi_{LL}}$
    - \* Note that we can divide both sides by  $u(w)$ :  $\varepsilon_L \leq \frac{\beta \pi_{LH} \varepsilon_H}{1 - \beta \pi_{LL}}$

- \* So, without growth in the size of the cake overtime, the decision rule is not a function of the size of the cake or the parameterization of the utility function.
- \* NOTE: this is not the case if the size of the cake is growing - as on one of your HW problems.

#### Some final tips on speeding up computations

- In Python, use `Numba` and it's just-in-time compilation to speed up loops
  - Otherwise, in Python or other higher-level languages, “vectorize” your code so that you are doing operations on arrays rather than through loops with element-by-element operations
- Consider parallel processing (e.g., to evaluate functions are different parts of your state space simultaneously)
  - The `dask` package for Python makes multiprocessing straightforward