Economics 2010c: Lecture 4 Incomplete Markets and Liquidity Constraints

David Laibson

9/13/2022

Outline:

1. Why does $E_t \Delta \ln y_{t+1}$ predict $\Delta \ln c_{t+1}$?

2. Incomplete markets: e.g., liquidity constraints

3. Numerical solution of a stationary problem with liquidity constraints

4. Comparison to "eat-the-pie" problem

5. Discrete numerical analysis (optional)

1 Why does $E_t \Delta \ln y_{t+1}$ predict $\Delta \ln c_{t+1}$?

- Welfare costs of smoothing are second-order (Cochrane 1989, Pischke 1995, Browning and Crossley 2001, Kueng 2015, Gabaix 2016)

- Leisure and consumption expenditure are substitutes (Heckman 1974, Ghez and Becker 1975, Aguiar and Hurst 2005, 2007, Stephens and Toohey 2020)

- Work-based expenses (see Ganong and Noel 2016)

- Households support lots of dependents in mid-life when income is highest (Browning 1992, Attanasio 1995, Seshadri et al 2006)

- Some consumers use rules of thumb: $c_{it} = \alpha Y_{it}$ (Campbell and Mankiw 1989, Thaler and Shefrin 1981, Gabaix 2016)

- Markets are incomplete and households are impatient (Deaton 1991, Carroll 1992, Kaplan and Violante 2014, Kaplan, Moll, and Violante 2018): these assumptions make the variance term time-varying ($V_t \Delta \ln c_{t+1}$).

- Markets are incomplete and households are present-biased (Laibson 1997, Harris and Laibson 2001, Shapiro 2005, Laibson, Maxted, Repetto, and Tobacman 2020, Laibson, Maxted, and Moll 2020): these assumptions violate the classical Euler Equation altogether.

2 Incomplete markets: e.g., liquidity constraints

Key assumptions (Zeldes, Carroll, Deaton):

1. Consumers can't sell their ex-ante stream of labor income. Instead, they face a borrowing limit: e.g. $c_t \leq x_t + L$.

   - $x_t + L$ is liquidity in period t; minimal 'savings' is $\min(x_t - c_t) = -L$.

   - This constraint matters whether or not it actually binds in equilibrium

2. Consumers are impatient $\rho > r$.

3. Utility is CRRA (and negative consumption is not allowed).

Predictions of models with these elements:

- Consumer decumulate if x_t is large, because $\rho > r$

- Consumers maintain a modest amount of liquid wealth, x_t, to buffer transitory income shocks (accordingly, these are called 'buffer stock' models)

Predictions continued:

- Consumption weakly tracks income at high frequencies (even predictable income)
  - when income temporarily falls, the liquidity constraint prevents households from borrowing to smooth consumption
  - when income temporarily rises, impatience causes households to splurge rather than smooth the windfall
- Consumption strongly tracks income at low frequencies (even predictable income)

We will revisit these predictions in coming lectures.

3 Application: Numerical solution

- Labor income iid, uniformly distributed on $[y_{MIN}, y_{MAX}] = \left[\frac{1}{2}, \frac{3}{2}\right]$.

- $c_t \leq x_t + L =$ liquidity

$$L \equiv \sum_{t=1}^{\infty} R^{-t} y_{MIN} = \frac{1}{R} \times \frac{y_{MIN}}{1 - 1/R} = \frac{y_{MIN}}{R - 1}$$

- $u(c) = \ln(c) = \lim_{\gamma \to 1} \frac{c^{1-\gamma} - 1}{1-\gamma}$

- Discount factor, $\delta = 0.9$

- Gross rate of return, $R = 1.05$

- Infinite horizon

- Solution method is numerical.

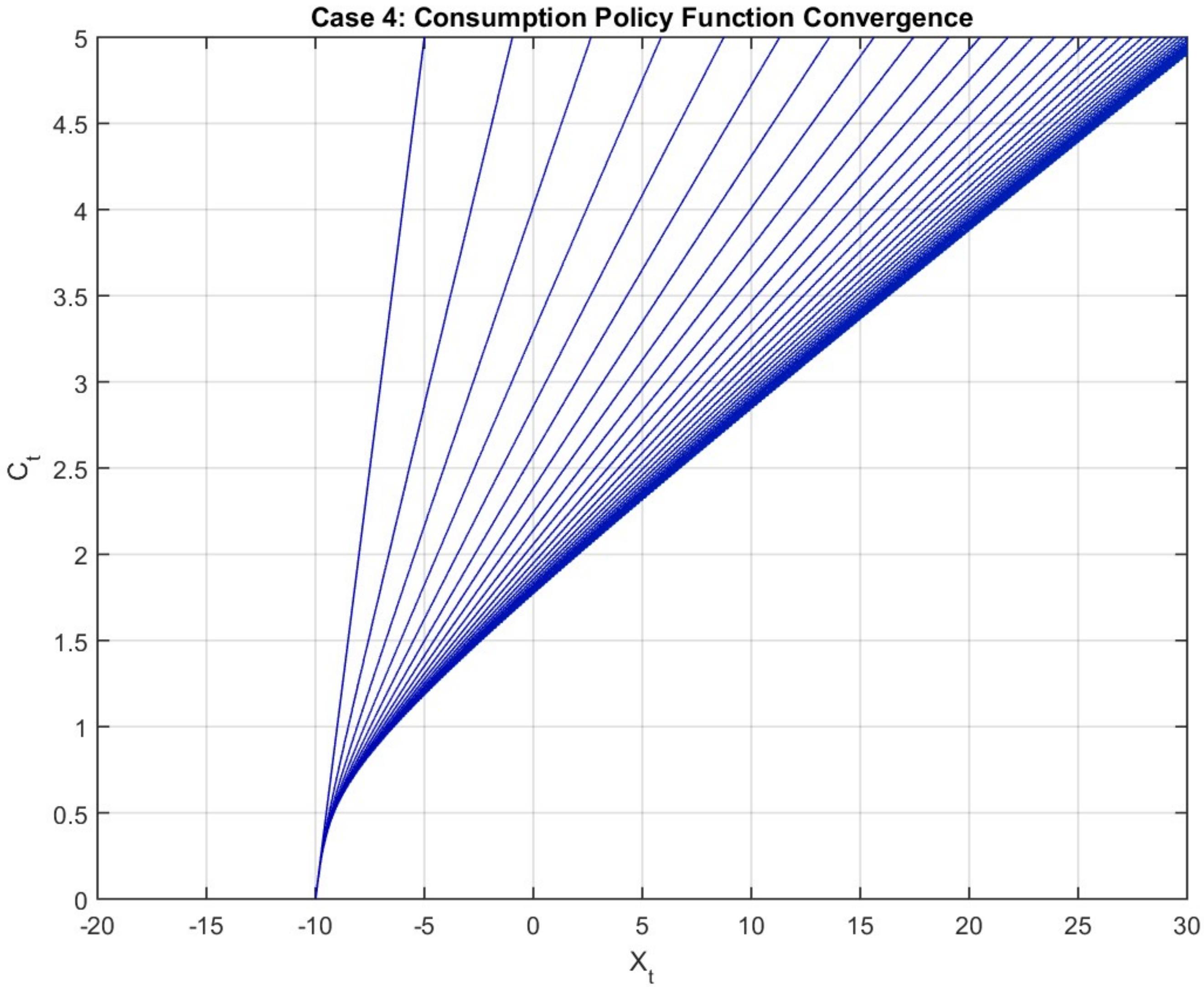
- Let

$$(Bf)(x) \equiv \sup_{c \in [0, x+L]} \{u(c) + \delta E f(R(x-c) + \tilde{y}_{+1})\} \quad \forall x$$

$$x_{+1} = R(x-c) + \tilde{y}_{+1}$$

- Solution given by: $\lim_{n\to\infty}(B^n f_0)(x)$.

- Iteration of Bellman operator is done on a computer (using a discretized state and action space) with $f_0(x) = 0$ for all x.



Consumption functions associated with 250 iterations of the Bellman Operator.

4 Eat the pie problem

Compare to a model in which the consumer can *securitize* her income stream. In this model, labor income can be transformed into a bond.

- If consumers have exogenous idiosyncratic labor income risk, then there is no risk premium and consumers can sell their labor income for

$$W_0 = E_0 \sum_{t=0}^{\infty} R^{-t} y_t.$$

- The dynamic budget constraint is

$$W_{+1} = R(W - c).$$

- Bellman equation for "eat-the-pie" problem:

$$v(W) = \sup_{c \in [0,W]} \{u(c) + \delta E v(R(W-c))\} \quad \forall x$$

- Guess the form of the solution.

$$v(W) = \left\{ \begin{array}{lll} \psi\frac{W^{1-\gamma}}{1-\gamma} & \text{if} & \gamma \in [0,\infty],\ \gamma \neq 1 \\ \phi + \psi \ln W & \text{if} & \gamma = 1 \end{array} \right\}$$

- Confirm that solution works (problem set).

- Derive optimal policy rule (problem set).

$$c = \psi^{-\frac{1}{\gamma}} W$$

$$\begin{aligned} \psi^{-\frac{1}{\gamma}} &= 1 - (\delta R^{1-\gamma})^{\frac{1}{\gamma}} \\ &= 1 - \delta \qquad \text{if } \gamma = 1 \end{aligned}$$

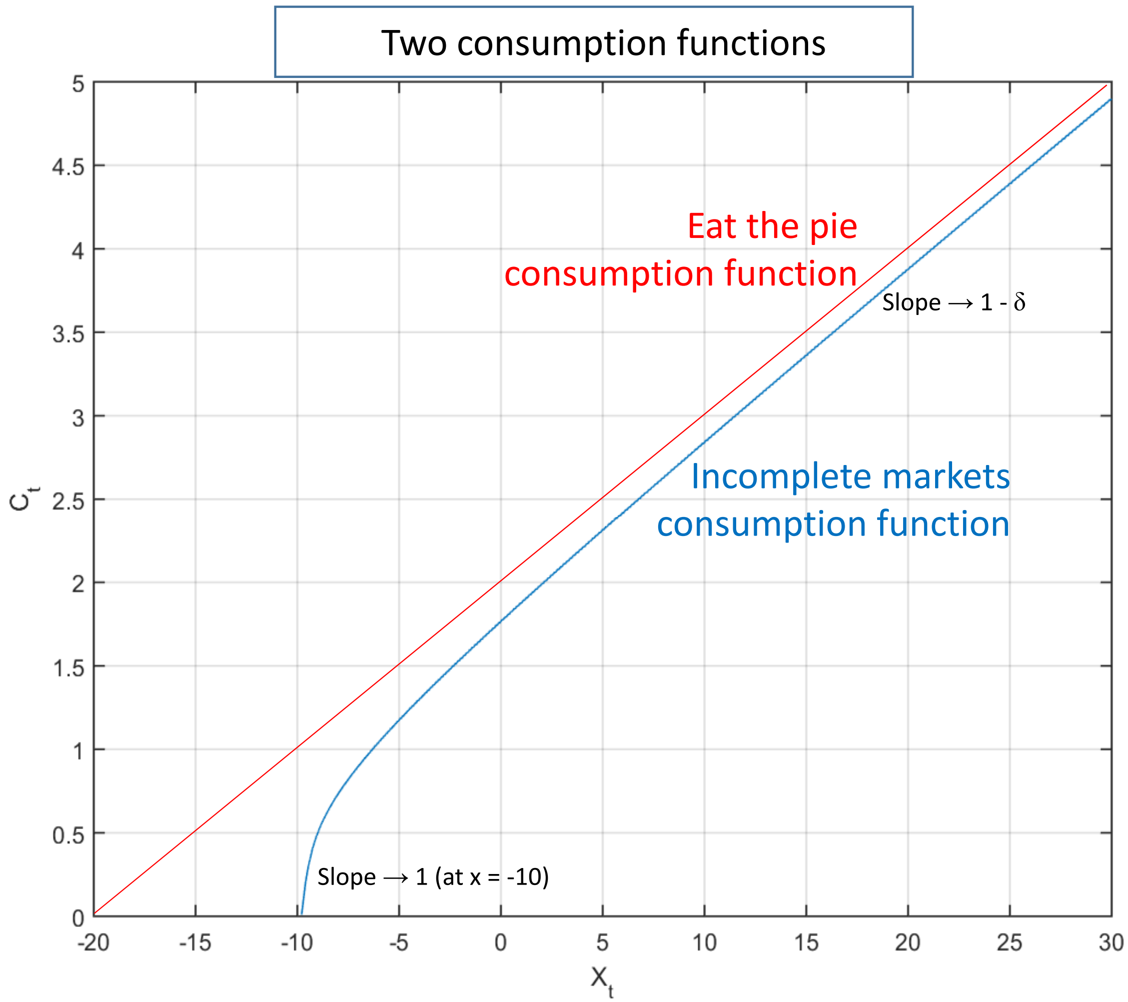
- Let's compare two *similarly situated* consumers at time period t:

  - a buffer stock consumer with cash-on-hand x_t, borrowing limit L, and a *non-tradeable* claim to all future labor income

  - an eat-the-pie consumer with cash-on-hand x_t and a *tradeable* claim equal to the expected value of the NPV of all future labor income; so the eat-the-pie consumer has current tradeable wealth

$$\begin{aligned} W_t &= x_t + E_t \sum_{s=1}^{\infty} R^{-s} y_{t+s} \\ &= x_t + \sum_{s=1}^{\infty} R^{-s} 1 = x_t + \frac{1}{R} \times \frac{1}{1 - 1/R} = x_t + \frac{1}{R-1} \end{aligned}$$

- Note that eat-the-pie consumption function is above optimal consumption function


Two consumption functions
Eat the pie
consumption function
Slope ⟶ 1 - δ
Incomplete markets
consumption function
Slope ⟶ 1 (at x = -10)
C_t
X_t
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-20
-15
-10
-5
0
5
10
15
20
25
30

5 Discrete numerical analysis (optional)

- basic idea is to partition continuous spaces into discrete spaces

- e.g., instead of having wealth in the interval [0, $5 million], we could set up a discrete space

$$0, \$1000, \$2000, \$3000, \ldots , \$5{,}000{,}000$$

- we could then let the agent optimize at every point in the discrete space (using some arbitrary continuation value function defined on the discrete space, and then iterating until convergence)

5.1 Example of discretization of buffer stock model

Continuous State-Space Bellman Equation:

$$v(x) = \sup_{c \in [0,x]} \{u(c) + \beta E v(R(x-c) + y_{+1})\}$$

We'll now discretize this problem.

- Consider a discrete grid of points
  $X = \{x_0, x_1, x_2, ..., x_{MAX}\}$.

- It's natural to set $x_0 = 0$ and x_{MAX} equal to a value that is sufficiently large that you never expect an optimizing agent to reach x_{MAX}.

- However, x_{MAX} should not be so large that you lose too much computational speed. Finding a sensible x_{MAX} is an art and may take a few trial runs.

- Consider another discrete grid of points
  $Y = \{y_0, y_1, y_2, ..., y_{MAX}\}$.

- Now, given $x \in X$, c is chosen such that

$$\begin{aligned} 0 &\leq c \leq x \qquad (1) \\ R(x-c)+y &\in X \quad \text{for all } y \in Y \qquad (2) \end{aligned}$$

- To make this last restriction possible, the discretized grids, X and Y, must be chosen judiciously.

- Define $\Gamma(x)$ as the set of feasible consumption values that satisfy constraints (1) and (2).

- So the Bellman Equation for the discretized problem becomes:

$$v(x) = \sup_{c \in \Gamma(x)} \{u(c) + \beta E v(R(x-c) + y_{+1})\}$$

An example of discretized grids X and Y.

Choose x_{MAX} to be divisible by Δ. Let

$$X \equiv \{0, \Delta, 2\Delta, 3\Delta, ..., x_{MAX}\}.$$

Let the elements of Y be multiples of Δ (e.g., $Y = \{13\Delta, 47\Delta\}$). Here I assume that the largest element in Y, y_{MAX}, is smaller than x_{MAX}.

Fix a cell $x \in X$. Let REM represent the REMainder generated by dividing x by $\frac{\Delta}{R}$. Then, $c \in \Gamma(x) \equiv$

$$\{REM, REM + \frac{\Delta}{R}, REM + \frac{2\Delta}{R}, ..., x - \frac{2\Delta}{R}, x - \frac{\Delta}{R}, x\}.$$

If $c \in \Gamma(x)$, then $x - c$ will be a multiple of $\frac{\Delta}{R}$, so

$$R(x - c) + y \in X$$

for all $y \in Y$.

Remark: For some (large) values of x, you will need to truncate the lowest valued cells of the $\Gamma(x)$ correspondence. Specifically, it must be the case that for every value of x,

$$R(x - \min\{\Gamma(x)\}) + y \leq x_{MAX}$$

for all $y \in Y$. This implies that

$$\min\{\Gamma(x)\} = \max\{REM, (x - \frac{x_{MAX}}{R}) + \frac{y_{MAX}}{R})\}.$$

5.2 Practical advice for Dynamic Programming

When using analytical methods...

- work with ∞-horizon problems (if possible)
- exploit other tricks to make your problem stationary (e.g., constant hazard rate for retirement)
- work with tractable densities (always try the uniform density in discrete time; try brownian motion and/or poisson jump processes in continuous time)
- minimize the number of state variables

When using numerical methods...

- approximate continuous random variables with discretized Markov processes
- coarsely discretize exogenous random variables
- densely discretize endogenous random variables
- use Monte Carlo methods to calculate multi-dimensional integrals
- use analytics to partially simplify problem (e.g., retirement as infinite horizon eat the pie problem)

- translate Matlab code into optimized code (C^{++})

- consider using polynomial approximations of value functions (Judd)

- consider using spline (piecewise polynomial) approximations of value functions (Judd)

- minimize the number of state variables (cf Carroll 1997)

5.3 Curse of dimensionality: travelling salesman

- must map route including visits to K cities
- job is to minimize total distance travelled
- state variable: a K-dimensional vector representing the K cities
- if the salesman has already visited a city, we put a one in that cell
- set of states is all K-dimensional vectors with 0's and 1's as elements

$$X = \prod_{i=1}^{K} \{0, 1\}$$

Bellman Equation:

$$v(x, \text{city}) = \max_{\text{city}'} \left\{ -d(\text{city}, \text{city}') + \delta v(x\prime, \text{city}') \right\}$$

Functional Equation:

$$(Bf)(x, \text{city}) = \max_{\text{city}'} \left\{ -d(\text{city}, \text{city}') + \delta f(x\prime, \text{city}') \right\}$$

How many different states $x \in X$ are there?

$$\sum_{n=0}^{K} \binom{K}{n}$$

This is a large number when K is large.

For example:

- $\binom{K}{n} = \frac{K!}{(K-n)!n!}$

- Let $K = 100$, $n = 50$,so $\binom{K}{n} = \frac{100!}{50!50!} = 10^{29}$

- And that's just one value of n

- To put this in perspective, a modern supercomputer can do a trillion calculations per second.

- So a supercomputer could go through one round of Bellman operator iteration in 10^{10} years.

Lesson:

- Even for seemingly simple problems the state space can get quite large.
- Work hard to limit the size of your state space.

- You will typically have state spaces that are in $\Re^l$.

- Suppose you had $l = 4$.

- Suppose you were modelling assets and you partitioned your state space into blocks of \$1000.

- Imagine that you bound each of your four assets between \$0 and \$1,000,000.

- Your state space has 1000^4 elements.

- So each round of Bellman iteration requires the computer to do 1000^4 computations.