Dynamic Optimisation

MATLAB AND MICRODATA PROGRAMMING GROUP

HILARY 2014
21 FEBRUARY

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

Choice over time

Dynamic problems have two aspects: stocks and flows.

- The <u>st</u>ate variable summarises <u>st</u>ocks
- The <u>c</u>ontrol variable is the variable being <u>c</u>hosen (ie flows)

$$U = \sum_{t=1}^{T} \beta^{t-1} u(c_t), \tag{1}$$

$$k_{t+1} = f(k_t, c_t). \tag{2}$$

A Dynamic Household

Attach functional forms to (1) and (2):

$$u(c_t) = \ln(c_t) \qquad\qquad f(k_{t+1}) = k_t - c_t$$

Then . . .

A Dynamic Household

$$\max_{\{c_t\}_1^T} \sum_{t=1}^{T} \beta^{t-1} \ln(c_t) \qquad \text{s.t.} \qquad \sum_{t=1}^{T} c_t + k_{T+1} = k_1 \qquad (3)$$

$$c_t \geq 0$$

$$k_t \geq 0.$$

MATLABbing it

We should be able to solve this problem by “direct attack” in MATLAB

- A function to maximise
- A vector of maximands
- A vector of upper and lower bounds
- A(n) (in)equality constraint
- our old friend `fmincon`

```matlab
function V = flowUtility(T,Beta,C)
    % flowUtility(T,Beta,C) takes T periods of
    % consumption of size C (a Tx1 vector), and
    % calculates the total utility of consumption
    % assuming an additively separable utility
    % function and discount rate β.

    t  =  [1:1:T];
    V  =  Beta.^(t-1)*log(C);
    V  =  -V;

return
```

Sensitivity

We have assumed a particular functional form, and values for input parameters

- Here we are imposing these, rather than recovering them
- Of course, we can re-solve the model based on alternative assumptions. . .
  - Alternative values of β
  - Alternative utility functions
  - Alternative forms of the flow equation (see chapter)

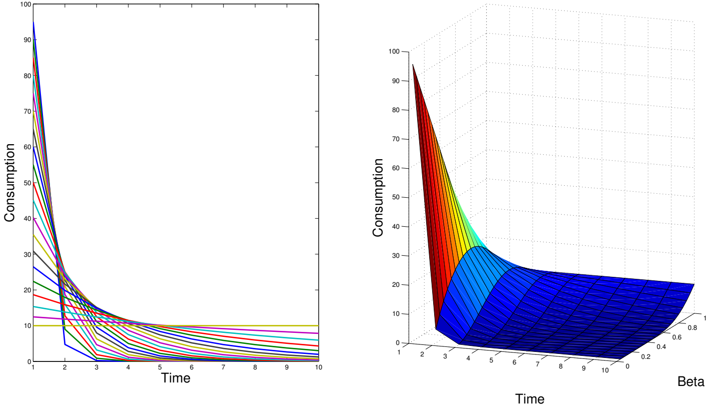

Figure: Sensitivity of Consumption to Discount Rate

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

The Bellman Equation

Generally when people speak about 'dynamic programming' in economics, they refer to the class of models solved using value function iteration.

- While solvers like `fmincon` are useful as a general outline, often we need more flexible methods of attack
- This is where the Bellman equation comes in handy
- Essentially, breaks down the problem into sequentially much smaller problems

The Bellman Equation

$$V(k_t) = \max_{c_t}\{u(c_t) + \beta V(k_{t+1})\} \qquad (4)$$

Iteration. . .

So, we can break this down into sub-problems:

$$
\begin{aligned}
V(k_T) &= \max_{c_T} \{u(c_T) + \beta V(k_{T+1})\} \\
V(k_{T-1}) &= \max_{c_{T-1}} \{u(c_{T-1}) + \beta V(k_T)\} \\
V(k_{T-2}) &= \max_{c_{T-2}} \{u(c_{T-2}) + \beta V(k_{T-1})\} \\
&\vdots \\
V(k_2) &= \max_{c_2} \{u(c_2) + \beta V(k_3)\} \\
V(k_1) &= \max_{c_1} \{u(c_1) + \beta V(k_2)\}
\end{aligned}
\qquad (5)
$$

Iteration II

Now, all we need is a place to start. . .

$$V(k_{T+1}) = 0 \;\; \forall \;\; k \tag{6}$$

MATLABbing it

We'll solve Bellman equations numerically with MATLAB

- Essentially, 'brute force' grid search
- Requires 'gridding' state variables (if not binary)
- Let's check out `backwardsInduc.m`

‘Memoization’

A brief final point here: this is computationally intense, but we can avoid a lot of repeated heavy lifting

- ‘Memoization’ (aka computer programming in ‘Nature’)
- This is something that comes in very handy when simulating and solving these problems

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

Outline

1. Direct Attack
2. The Bellman Equation
3. Uncertainty

Uncertainty

What we've seen so far is actually remarkably flexible.

- Generalises quite simply (in theory) to multiple state and control variables
- Though in practice, curse of dimensionality
- Perhaps the only major thing we're missing is stochastic elements

The Bellman Equation

$$V(k_t) = \max_{c_t}\{u(c_t) + \beta\mathbb{E}[V(k_{t+1})]\} \tag{7}$$

Decisions Under Uncertainty

So, now the decision must framed in terms of consumption now and *expected* consumption in the future.

- In this case, the backwards iteration step is similar
- However, the iterating forwards to solve the model depends upon progressive realisations of shocks
- If time: `finiteStochastic.m`, `simulateStochastic.m`

Simulations

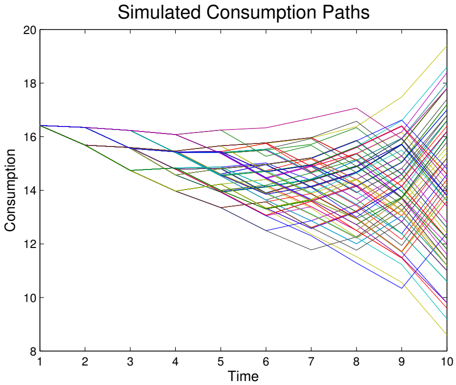

Figure: Simulated Consumption in a Stochastic Model

Summary

This week:

- Finite horizon dynamic optimsation
- Bellman equations
- A little bit of model simulation

Next week:

- Infinte horizons
- Using Bellman again
- Estimation!!

Dynamic Optimisation

MATLAB AND MICRODATA PROGRAMMING GROUP

HILARY 2014
21 FEBRUARY