November 9th 2009 - Lecture Fourteen

Econ205A – Macro 20091109

Outline – Quiz. The Dynamic Programming Problem and Bellman Equations Intro. Setting up finite horizon Bellman equation. Recipe for Solving Infinite Horizon Dynamic Programming Problems. Two examples – social planner growth model & the growth model with habit formation.

Quiz Three

Given this utility function,

$$u(c, 1-h)$$

Write down the social planner's problem and derive a pair of Conditions that characterize the optimal solution

Quiz Solution – The following answer is a bit dirtier than you want for a problem set or test. I am just writing down verbatim my answer to the quiz question. I also did weird stuff by setting $h = 1,$ which you don't want to do. The score was 10/10

The Problem

$$\max_{c,h} \sum_{t=0}^{\infty} \beta^t u(c, 1-h)$$
$$s.t. \quad c + i \leq y$$
$$y = f(k, h)$$
$$k_{t+1} = (1-\delta)k + i$$
$$c + k_{t+1} - (1-\delta)k = i$$
$$c \geq 0, \qquad 0 \leq h \leq 1, \quad k \geq 0$$

There is no disutility to labor in this model, but I'll set $h = 1$.

We are looking for an efficient allocation for $\{c\}, \{i\}, \{y\}, \{k\}, \{h\}$

The Lagrangian is

$$\mathcal{L}(c, h, k, \lambda) = u(c, 1-h) + \lambda[f(k, h) - c - k_{t+1} + (1-\delta)k)$$

FOCs

$$(1)\ \mathcal{L}_c = \beta^t u_1' - \lambda = 0$$
$$(2)\ \mathcal{L}_h = -\beta^t u_2' + \lambda f_2'(k, h) = 0$$
$$(3)\ \mathcal{L}_\lambda = f(k) - c - k_{t+1} + (1-\delta)k$$
$$(4)\ \mathcal{L}_k = \lambda_{t+1} + \lambda[f'(k) + (1+\delta)]$$

Four equations & four unknowns. I am sure we can solve this problem – given enough time….

Note that on $f'(k)\ and\ f_2'(k, h)$ – I was going to set $h = 1$ after we impose SS. And $f'(k) = f_1'(k, h)$

$$(1) \Rightarrow \beta^t u_1' = \lambda \qquad (4) \Rightarrow \frac{\lambda_{t+1}}{\lambda_t} = f'(k) + (1+\delta)$$

$$(1)\ \&\ (2) \Rightarrow \frac{u_2'}{u_1'} = f_2'(k, h) \quad (1)\&(4) \Rightarrow \frac{u_1'(c_{t-1}, 1-h_{t-1})}{\beta u_1'(c_t, 1-h_t)} = f'(k_t) + (1+\delta)$$

Given functional forms, I have no doubt that I could solve for SS values of c, I, k….

I also noted that there exists a solution at $c = 0, h = 0, and\ k = 0.$ But it's not interesting.

End of my Solution to the Quiz

Dynamic Programming – the problem

Last time we have introduced the methodology of dynamic programming. We'll continue with that today.

The basic problem we deal with is,

$$\max_{s_t, a_t} \sum_{t=0}^{T} \beta^t \, u(s_t, a_t)$$

$$s.t. \;\; 0 \leq \psi(s_t, a_t) \quad \& \quad s_{t+1} = h(s_t, a_t)$$

$o \leq \psi(s_t, a_t)$ is **the feasibility constraint**. This means that the control variable(s), a, that you choose are feasible given the subset of possible $a's$ you're choosing from.

$s_{t+1} =$ **The law of motion for the state variable**. It tells you, given your state today and the choice that you make for a, what will your state variable(s) be tomorrow.

There are two ways to solve this problem.

1) We can treat this like any old problem. Consider it a problem with many choice variables and constraints. There's be $T + 1$ constraints for one – and for the LOM there are T constraints. We'd just plug those into our old system and solve.

Dynamic Programming – solution outline

But we can also solve the problem backwards, recursively.

2) You can write down a simple problem by supposing that you've come into the final period with state s_t:

Solving the Problem at the Terminal Period, time = T

Imagine you enter T with $s_T, then$

$$\max_{a_T} u(s_t, a_t)$$

$$s.t. \quad 0 \leq \psi(s_T, a_T)$$

This becomes a very easy problem to solve. A micro type problem. You're maxing one variable subject to a constraint.

The solution is $\Pi_0(s) = a_T$

$this\ is\ an\ optimal\ policy\ function$

How did we start this, we defined the value $V_0(s) = u(s, \Pi_0(s))$

Using this solution we started working backwards – we defined the value that you get with zero periods remaining, given your state s as being equal to the utility you get with state s following the optimal policy in the terminal period. V_0 is just the maximized value of this solution. So if you take the solution you get solving this problem and plug that into the utility function, you get the maximized value, V_0.

Notation Note – the subscript in $V\_(s)$ refers to the number of periods left. $V_0 =$ zero periods left. $V_1 =$ one period left. $V_T = T$ periods left.

Solving the Problem at Period $T-1$

Doing this we have completely characterized what your choices are if you enter the last period with any state. We now want to work on finding a generalized solution to the problem any period. Let's do that by solving for period $T-1$.

$$Current\ Period\ is\ T-1, with\ s = s_{T-1}$$

$$V_1(s) = \max_{\substack{a_{T-1}, s_T \\ (a,s')}} u(s,a) + \beta V_0(s_T)$$

$$s.t.\ 0 \le \psi(s_t, a_t) \quad \& \quad s' = h(s,a)$$

- Maxing over $(a_{T-1}, s_T) \rightarrow (a, s')$. you are maxing over a_{T-1} & $implicitely\ over\ s_T$ – where s_T is state in the next period.
- $u(s,a)$ =current period utility
- $\beta V_0(s_T)$ = the discounted value of the state you bring into the next period.

Subject To

- $0 \le \psi(s_t, a_t)$ = feasibility constraint
- $s' = h(s,a) \rightarrow$ Law of Motion of State variable

Notation Note – Maxing Over $(a_{T-1}, s_T) \rightarrow (a, s')$. Firstly, you are maxing over a_{T-1} & $implicitely\ over\ s_T$ – where s_T is state in the next period. Notationwise, we shall remove time subscripts by using ' (apostrophe) to note what variables are the next period's (all your state variables). Those variables without apostrophes (all your control variables) don't have the '. This notation isn't required – you can use time subscripts – but Bellman equations are supposed to be time-ignorant, they don't need subscripts. And if anything, removing time subscripts it helps you keep your work a lot more organized.

Summary of the Set-up - V_1 is defining the maximum value you are able to attain from period $T-1$ onward if you enter the period with state s. Simple, a problem with two variables and two constraints.

Solving this, you'll find a optimal policy function

$$\Pi_1(s)$$

Solving the Problem at Period 0

Imagine that you keep doing this over and over again until you reached period zero. At period zero we want to find the optimal value given s,

$$V_T(s) = \max_{a,s'} \ u(s,a) + \beta\, V_{T-1}(s')$$

$$s.t.\ 0 \le \psi(s,a) \quad \& \quad s' = h(s,a)$$

This is saying that in the first period (the sub-T in V_T), entering with some values for your state variables (the (s) in $V_T(s)$), your maximum possible utility ($V_T(s)$) is your initial utility ($u(s,a)$), plus the best that you can do starting next period ($\beta\, V_{T-1}(s')$), entering next period with s'.

If you were to write out all of these V_t … it's going to give you the sum all the discounted period functions.

This gives us the policy function:

$$solution: \Pi_T(s)$$

If you carry this all the way out we now have a sequence of $T+1$ one-dimensional optimization problems.

The claim is that you can solve them for $V_0, V_1 \ldots V_T$ & $\Pi_0, \Pi_1 \ldots \Pi_T$ and you will have solved the original problem.

Sometimes this is an easier method to solving the problem than tackling the whole large scale problem.

It's easier to turn this method into a computer aided solution.

Infinite Horizon Problem

Now that method is great for a finite problem. But what are we going to do with an infinite problem?

Well, we certainly can't solve this problem backwards. We can't start at period ∞.

Luckily for us the problem is now exactly the same no matter what period you start ini – because no matter what day t is at, you'll always have an infinite horizon ahead of you. With this method time doesn't matter. Your decisions do not depend on how much time there is left. Time is irrelevant in an infinite horizon problem.

The Sequence Problem

$$\max_{s_t a_t} \sum_{t=0}^{\infty} \beta^t u(s_t, a_t)$$

$$s.t.\ 0 \leq \psi(s_t, a_t) \quad \& \quad s_{t+1} = h(s_t, a_t)$$

The Bellman Equation

Following this procedure we can write down the Bellman equation we desire.

$$V(s) = \max_{a,s'} u(s, a) + \beta V(s')$$

$$s.t.\ 0 \leq \psi(s_t, a_t) \quad \& \quad s' = h(s_t, a_t)$$

$V(s)$ – The value you get for having a given state s,
is equal to your optimal choice choosing control and implicitly the state next period,
equal to your period utility function plus the discounted value that you get for the state next period.
Subject to: Feasibility & the Law of Motion of your state variable(s)

Notation Note – On $V_T(s) \rightarrow V(s)$. In the finite horizon case we had a subscript on $V_T(s)$ labeling value functions separately depending on how long the horizon ahead was. Now we don't need to keep track of separate V for each time

Great – we have nice notation for a problem that is really hard to solve. It's a functional equation that depends on itself – and we need to solve for a function. Thanks Bellman.

The next several lectures will be on how to solve objects in this.

One nice feature is that we don't have time subscripts anymore. It doesn't matter what period we're in.
Also we can use the guess and check method. Say, guess V_T - plug it into our problem and see what happens to $V\ as\ t \rightarrow \infty$ - Shooting. What this means is that if you solve for the finite horizon problem in long enough a time horizon – that solution will be very close to the infinite horizon problem.

Eventually we'll be solving all these problems in the computer – the next many quarters – this method provide the algorithm to follow. An intro to the methodology.

Recipe for Solving Infinite Horizon Dynamic Programming Problems.

We now can go over the steps to the solution.
Also going over the mathematical rules we need to follow using this method.
Plus steps to analyze dynamic models

1) Write down the problem as a sequence problem (SP)

SP ≠ Social Planner. This is what we've just done above. To maximize sequences of control and state variables that maximize some objective function subject to some constraints.

2) Choose state variables,
and state space,
then write down the recursive problem, (the Bellman Equation)

A little vague today, we're just writing down the steps now and talk about them a little bit. We'll get back to how exactly to do this later and the mathematical conditions required for this method to work. This will follow in many ways the treatment of Stokey Lucas & Prescott, Chapter Two (I think).

3) Check that some basic conditions hold.

This is again pretty vague. We need sufficient conditions so that dynamic programming (or the Bellman equation) is a legitimate solution method. Hmmmmm… We need to check that in formulating the problem this way the solution we get is a sufficient condition for optimization – there are some properties that we'll need.

4) Check some other conditions to get properties of the

The feasibility and law of motion of capital. The state variable is the variable that defines the state. The state space is where that variable is allowed to be.
In the past we've said; given some state variable x, its an element of some space X. Thus it will be required that there are restrictions on the state space for dynamic programming to work.

I guess you can say your state variables are bounded above and below each period. Bounded between zero consumption and full consumption $(1-\delta)k \leq k_{t+1} \leq (1-\delta)k + f(k)$.

also Conditions that Will Ensure Sufficient Properties of the Solution

- Monotonicity (the optimal policy function) If the state is capital and the control is consumption, do you consume more over time?
- Concavity -
- Differentiability

These are properties you want your solutions to have.

5) Solve for the Euler Equation

The Euler Equation – the first order equations that come out of the model are the empirical content of the model. The meat. What you eat. Why we're here.

6) Look for Steady State and Examine Transitional Dynamics (local stability).

7) Comparative Dynamics

Ask how a change in parameters affects the solution

Here is the basic outline of where we are headed.
The first thing to do is get practice going from step one to step two – examples.

Example of this process – Setting up a Bellman steps one and two

1) Social Planner's problem from the growth model.

$$\max \sum_{t=0}^{\infty} \beta^t u(c_t)$$

$$s.t. \quad 0 \le c_t \le f(k_t) + (1-\delta)k_t$$

$$k_{t+1} = f(k_t) + (1-\delta)k_t - c_t$$

$$k_0 \; is \; given$$

We can ask, is this problem recursive? Does it fit into the basic formulation of states and controls with the sequence problem that we had? Can all these variables translate into $s's$ & $a's$, state and controls, feasibility and laws of motion?

Yes

$state$:	s_t	$\rightarrow$	k_t
$control$:	a_t	$\rightarrow$	c_t
$preferences$:	u	$\rightarrow$	$u(c_t)$
$feasability$:	$0 \le \psi(s_t, a_t)$	$\rightarrow$	$0 \le c_t \le f(k_t) + (1-\delta)k_t$
$law\ of\ motion$:	$s_{t+1} = h(s_t, a_t)$	$\rightarrow$	$k_{t+1} = f(k_t) + (1-\delta)k_t - c_t$

We now want to write a Bellman equation. The state is k, so we have $V(k) = \cdots$
We maximize over the state and control variables, c, & k'. Consumption today and capital tomorrow.

$$V(k) = \max_{c,k'} u(c) + \beta V(k')$$

$$s.t. \quad 0 \le c_t \le f(k_t) + (1-\delta)k_t$$
$$\& \quad k_{t+1} = f(k_t) + (1-\delta)k_t - c_t$$

If you'll notice we haven't brought over that "$k_0 \; is \; given.$" Why not? we don't need it because you don't need an initial condition.

Now, what are we solving for? We are solving for $\Pi(k)$, a policy function, and optimal decision rule that is a function of k. What is V? it's the optimal total future value of someone with state k. We are solving for any k. It doesn't matter what your initial k because our solution will be generalize for any initial value of k.

Solving for V delivers the value for every k, and the optimal policy for every k. Solving for this solves for every possible contingency.

The solution gives the equation

$\Pi(k)$ = the optimal choice of c given any k

Example 2 – Growth Model with Habit Formation
Another example of just steps one & two, setting up the Bellman correctly

$$Growth\ model\ with\ habit\ formation$$
$$\max \sum_{t=0}^{\infty} \beta^t u(c_{t-1}, c_t)$$
$$s.t. \quad 0 \le c_t \le f(k_t) + (1-\delta)k_t$$
$$k_{t+1} = f(k_t) + (1-\delta)k_t - c_t$$
$$k_0 \ \&\ c_{t-1}\ given$$

State Variable – it can't be just k_t now. The state variable also includes c_{t-1}

$$s_t = \begin{bmatrix} k_t \\ c_{t-1} \end{bmatrix}$$
$$s_{t+1} = \begin{bmatrix} f(k_t) + (1-\delta)k_t - c_t \\ c_t \end{bmatrix} = s'$$

The state has all the information that an agent needs to make a decision today. That is definitely k_t & c_{t-1}

$$V(c_{t-1}, k_t) = \max_{c_t, k_{t+1}} u(c_{t-1}, c_t) + \beta V(c_t, k_{t+1})$$

$$s.t. \quad 0 \le c_t \le f(k_t) + (1-\delta)k_t$$
$$k_{t+1} = f(k_t) + (1-\delta)k_t - c_t$$

Next time we'll go over an example and go over all the technical details needed to solve problems like these.

See Stockey, Lucas & Prescott textbook for info too.