Dynamic Optimisation, Estimation

MATLAB AND MICRODATA PROGRAMMING GROUP

HILARY 2014
21 FEBRUARY

Outline

Outline

Infinite Horizons

With *finite* horizon problems, we could use V_T to seed solution

- We could iterate backwards from the terminal value function, and use this to recursively solve models
- But, in many cases there won't be some obvious end point...
- Fortunately we can still use Bellman's technique if we can find some optimal policy 'forever'
- What's more, the way we solve this is also iterative

Solving Infinite Problems...

$$V(k) = \max_{c}\{u(c) + \beta V(\tilde{k})\} \tag{1}$$

cf:

$$V_t(k_t) = \max_{c_t}\{u(c_t) + \beta V_{t+1}(k_{t+1})\}$$

Finding an Optimal Policy Forever

In (1), $u(c)$ is quite clear, but calculating $V(k)$ is the tricky part

- Value function is the same on both sides of the equation
- So, solving this involves finding a fixed point
- Bellman shows that quite conveniently, starting with any $V_j(k)$, we will eventually iterate onto our stationary $V(k)$

Trying One Out

$$\max_{\{c_t\}_{t=1}^{\infty},\{k_t\}_{t=2}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} \ln(c_t) \qquad \text{subject to} \qquad \begin{aligned} k_{t+1} &= \theta k_t^{\alpha} - c_t \\ c_t &\geq 0 \\ k_t &\geq 0. \end{aligned} \tag{2}$$

Trying One Out

Fortunately this *also* gives an analytical solution!

$$V(k) = \frac{\alpha}{1-\beta\alpha} \ln k + F \tag{3}$$

$$c(k) = \theta k^{\alpha}(1-\beta\alpha). \tag{4}$$

If you need convincing, check out appendix to notes. . .

Iterating to the Value Function

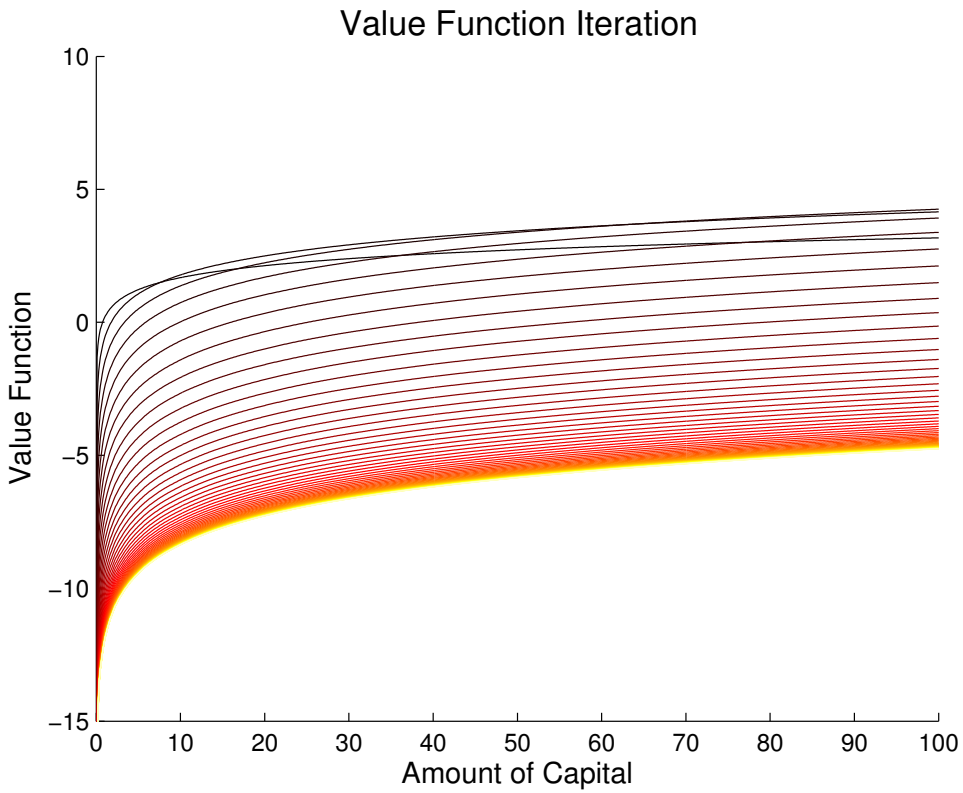
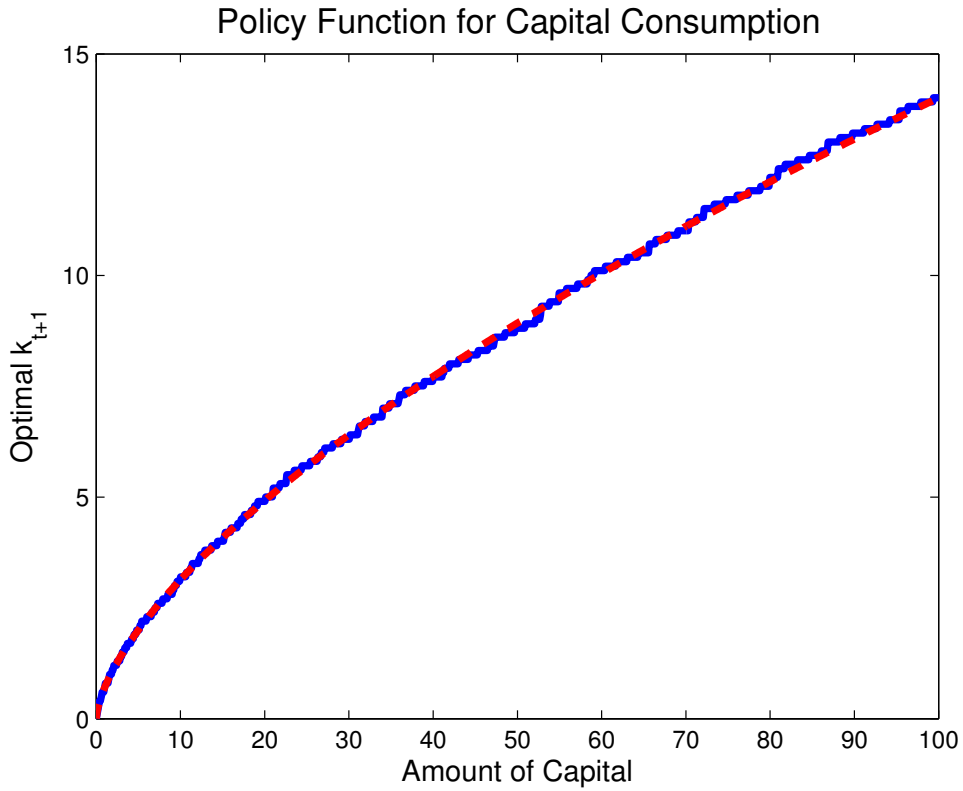


Figure: Convergence to the Numerical Value Function

The Solution

And then, with value function $V(k)$ in hand it's just a matter of determining the optimal policy— the "policy function"—$c(k)$.



Outline

Outline

The Howard Improvement Algorithm

Value function iteration can be quite slow and computationally intensive

- There are a number of ways to speed up these problems, including 'policy function iteration'
- Our value function iteration takes between 66-135 iterations depending upon ε
- In policy function iteration we follow each proposed solution forever, rather than just for one period

The Algorithm

1. Based upon V_j, determine optimal consumption for each k, giving a proposed 'policy function', $c_j(k)$
2. Calculate the payoff associated with this policy function, $u(c_j(k))$
3. Calculate the value of following this policy function forever, V_{j+1}
4. If $||V_{j+1} - V_j|| < \varepsilon$ stop, or else return to step (i) for another iteration

Remaining Bottlenecks

In step (3):

$$\begin{aligned} V_j &= u(c_j(k)) + \beta Q_j V_j \\ \Rightarrow V_j &= (I - \beta Q_j)^{-1} u(c_j(k)), \end{aligned} \tag{5}$$

Policy Function Iteration

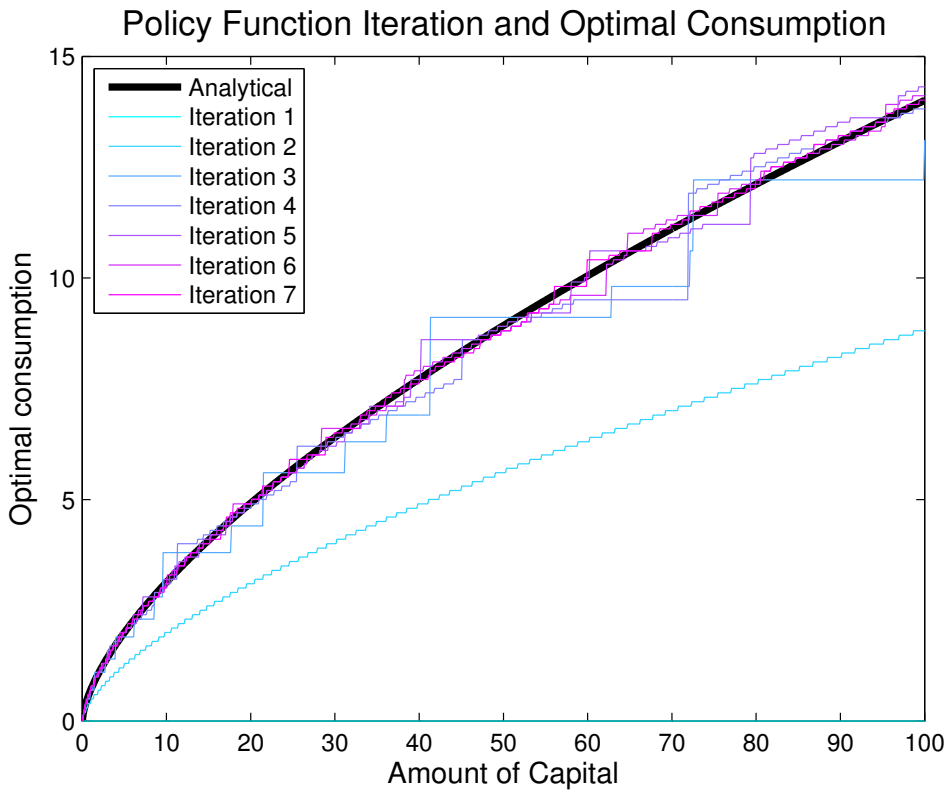


Figure: Convergence to the True Policy Function

Outline

Outline

Changing Viewpoints Entirely...

Forward versus **Inverse** Problems

Generalised Method of Moments

Our old friend...

Population moments:

$$E[\boldsymbol{X}\boldsymbol{\varepsilon}] = E[\boldsymbol{X}(y - \boldsymbol{X}\boldsymbol{\beta})] = \mathbf{0}. \tag{6}$$

And by analogy:

$$\boldsymbol{m} = \frac{1}{N}\left[\sum_{i=1}^{N} \boldsymbol{X_i}(y_i - \boldsymbol{X_i}\boldsymbol{\beta})\right] = \mathbf{0}. \tag{7}$$

Outline

Outline

Estimating Dynamic Models

So, going from moments to MATLAB isn't too difficult...

- The challenge is in knowing which moments to fit!
- Let's return to the example from last week with stochastic elements
- Our moments can be based on $\mathbb{E}[\varepsilon_t] = 0$.

Stochastic Dynamic Model

Remember:

$$\max_{\{c_t\}_{t=1}^{T}} \sum_{t=1}^{T} \beta^{t-1} u(c_t) \qquad \text{subject to} \qquad k_{t+1} = f(k_t, c_t) + \varepsilon_{t+1}. \quad (8)$$

Moments

$$\mathbb{E}[k_{t+1} - f(k_t, c_t)] = 0 \tag{9}$$

$$\mathbb{E}\left[\frac{u'(c_t)}{\beta u'(c_{t+1})} - f'(k_t)\right] = 0. \tag{10}$$

Moments II

$$\mathbb{E}[k_{t+1} - \theta(k_t - c_t)^{\alpha}] = 0 \quad (11)$$

$$\mathbb{E}\left[\frac{c_{t+1}}{\beta c_t} - \alpha\theta(k_t - c_t)^{\alpha-1}\right] = 0. \quad (12)$$

Dynamic Optimisation, Estimation

MATLAB AND MICRODATA PROGRAMMING GROUP

HILARY 2014
21 FEBRUARY