Lecture 1

Continuous Time Models

Econ 2713A: Computation

University of Pittsburgh, 2018

Baseline

For comparison, let's consider the neoclassical growth model social planner in discrete time

$$V(k) = \max_i \{u(f(k) - i) + \beta V((1-\delta)k + i)\}$$

This has the optimality conditions

$$\begin{aligned} u'(c) &= \beta V_k(k') \\ V_k(k) &= u'(c) f'(k) + \beta(1-\delta) V_k(k') \end{aligned}$$

Combining these yields

$$\frac{u'(c)}{u'(c')} = \beta \left[f'(k) + (1-\delta) \right]$$

Solution

In steady state, we have

$$f'(k^*) = \frac{1-\beta}{\beta} + \delta$$
$$c^* = f(k^*) - \delta k^*$$

We can solve the dynamics with value function iteration

$$V^{r+1}(k) = \max_i \{u(f(k) - i) + \beta V^r((1-\delta)k + i)\}$$

Then hope for convergence

$$\lim_{r \to \infty} V_r = V^*$$

Continuous Limit

Now let's take this to the limit where timesteps Δ are small

$$V(k) = \max_i \{\Delta u(f(k) - i) + \exp(-\rho\Delta) V((1 - \Delta\delta)k + \Delta i)\}$$

The optimality condition here is

$$\begin{aligned} & \Delta u'(c) = \Delta \exp(-\rho\Delta) V_k((1 - \Delta\delta)k + \Delta i) \\ \lim_{\Delta \to 0} \to \; & u'(c) = V_k(k) \end{aligned}$$

Similarly, the envelope condition is

$$\begin{aligned} & V_k(k) = \Delta u'(c) f'(k) + \exp(-\rho\Delta)(1 - \Delta\delta) V_k(1 - \Delta\delta)k + \Delta i) \\ \lim_{\Delta \to 0} \to \; & (\rho + \delta) V_k(k) = u'(c) f'(k) + (i - \delta k) V_{kk} \end{aligned}$$

Euler Equation

We can take derivatives and substitute to find

$$\left[-\frac{u''(c)c}{u'(c)}\right]\frac{\dot{c}}{c} = f'(k) - (\rho + \delta)$$
$$\dot{k} = f(k) - c - \delta k$$

The limiting form of the original value function equation is

$$\rho V(k) = u(f(k) - i(k)) + (i(k) - \delta k)V_k(k)$$
$$u'(f(k) - i(k)) = V_k(k)$$

Rediscretization

No we can avoid the optimization loop of the discrete case

Ironically, we actually go back to discete form to compute

$$V^{r+1}(k) = \Delta u(f(k) - i(k)) + \exp(-\Delta\rho)\left[V^r(k) + \Delta(i(k) - \delta k)V_k^r(k)\right]$$
$$u(f(k) - i(k)) = \exp(-\Delta\rho)V_k^r(k)$$

This can be seen to have the continuous limit seen earlier

However, we still need to compute numerical derivatives

$$V_k(k) \approx \frac{V(k+\varepsilon) - V(k)}{\varepsilon}$$

Upwinding

Lurking in the background is still the dreaded… transversality condition

In essence, our transversality condition is that we should eventually reach steady state

To account for this, we point our numerical derivatives in the direction of steady state (i.e., $\dot{k}$)

$$V_k^+(k) = \frac{V(k+\varepsilon) - V(k)}{\varepsilon} \quad \text{and} \quad V_k^-(k) = \frac{V(k-\varepsilon) - V(k)}{\varepsilon}$$
$$V_k(k) = q(k)V_k^+(k) + (1-q(k))V_k^-(k)$$

where $q(k) = \Phi(\dot{k}(k)) = \Phi(i(k) - \delta k) \in (0,1)$ and Φ is some sigmoid function like the normal CDF

Numerical Derivatives

Ultimately we will be performing these operations on a finite grid: k^j for $j \in \{0, \ldots, N-1\}$

Analogously, we will have discrete value function points V^j, so the derivative is approximately

$$V^j_{k+} = \frac{V^{j+1} - V^j}{k^{j+1} - k^j} \quad \text{and} \quad V^j_{k-} = \frac{V^j - V^{j-1}}{k^j - k^{j-1}}$$

$$V^j_k = q_j V^j_{k+} + (1 - q_j) V^j_{k-}$$

Letting $\mathbf{v} \equiv \left(V^0, \ldots, V^{N-1}\right)$ this then yields a linear equation

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{v}$$

Note that the matrix $\mathbf{A}$ in this case will be extremely sparse

Sparse Matrices

Sparse matrices are those that contain mostly zero elements

They can be represented more compactly on a computer if instead of storing a value for each (i, j) pair, we store a list of triplets (i, j, v) describing the location and value of all non-zero elements

In practice we actually store three separate vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{v}$ of common length

Sparse solvers can be quite a bit faster and are much more memory efficient

General Algorithm

Here is the general procedure for solving this optimization problem

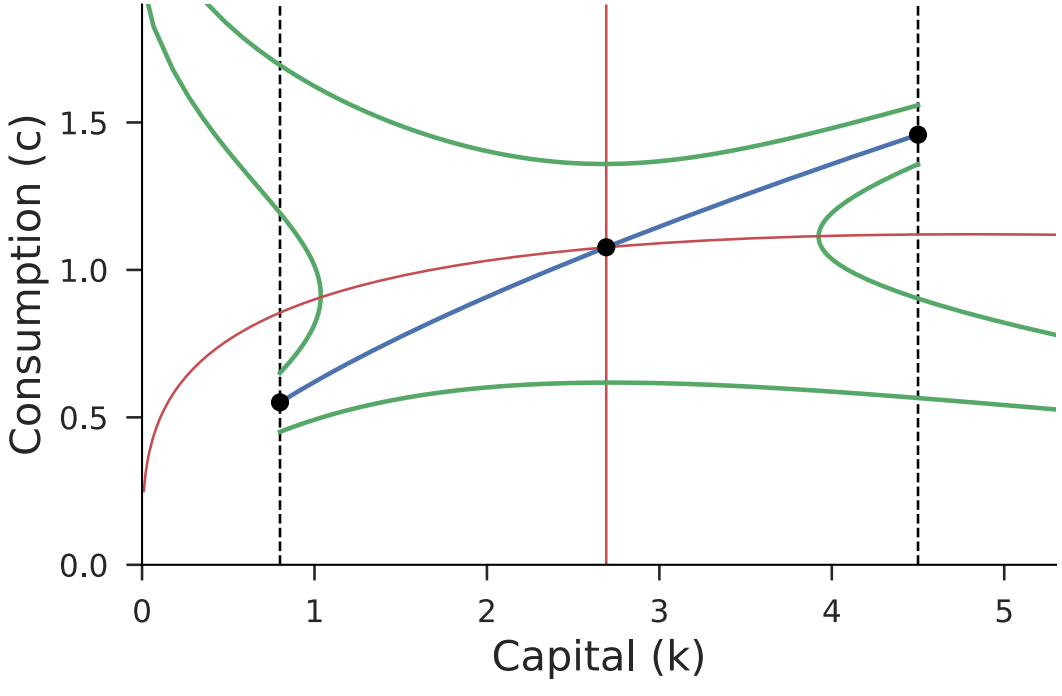
First, initialize a guess for investment $i_0^j = s(k^j)^\alpha$, then repeat the following until convergence:

1. Construct $\mathbf{b}_r$ and $\mathbf{A}_r$ using $\mathbf{i}_r$ and solve the linear system $\mathbf{b}_r = \mathbf{A}_r \cdot \mathbf{v}$ to get $\mathbf{v}_r$
2. Use $\mathbf{v}_r$ to find updated investment i_{r+1}^j with equation $u(f(k^j) - i^j) = V_k^j$

Shooting Method

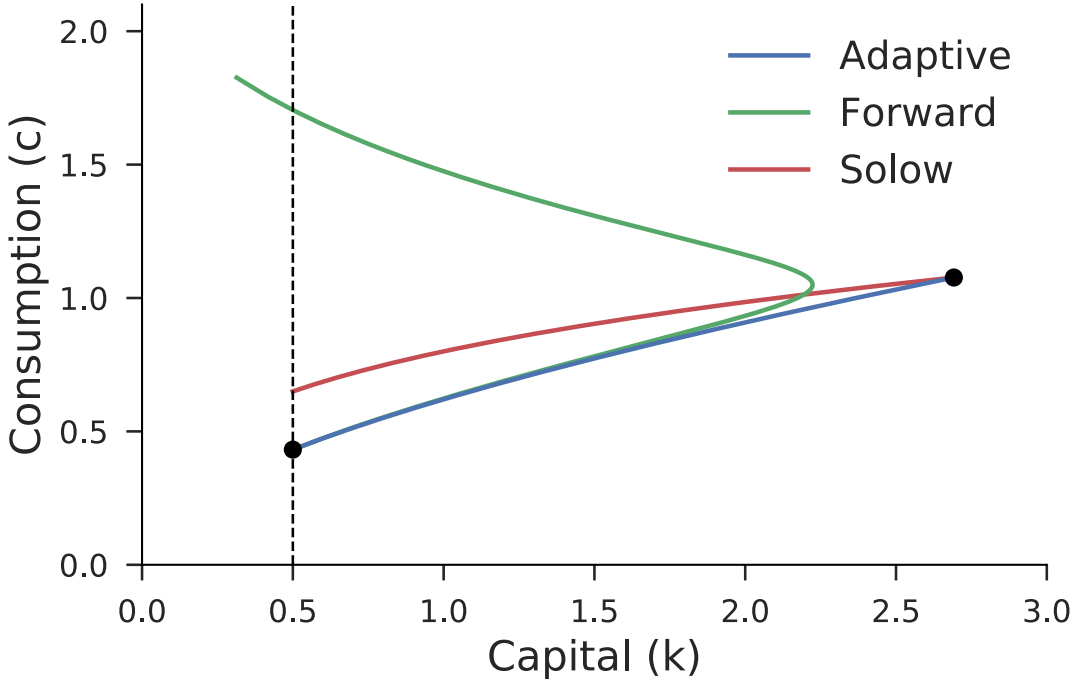
Can also solve this problem using the laws of motion for $\dot{k}$ and $\dot{c}$

Given a value for $k(0)$ and a guess for $c(0)$ we can simulate a path



Instability

Transversality says we should end up at steady state, however because the system is unstable, we will never reach it for any real values: errors compound with each step



Gonna Go Back In Time

Linearizing a system around steady state

$$\frac{d(x - x^*)}{dt} \approx J^* \cdot (x - x^*)$$

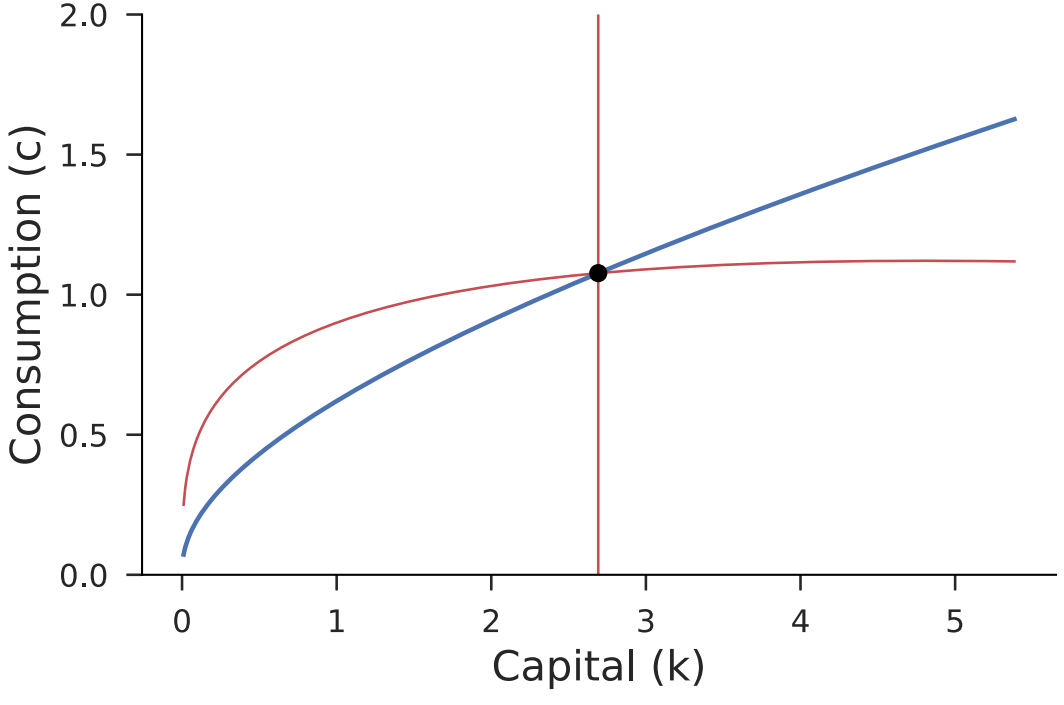
In our case we have

$$J^* = \begin{bmatrix} \rho - n & -1 \\ \frac{c^* f''(k^*)}{\varepsilon_u(c^*)} & 0 \end{bmatrix}$$

Saddle path dynamics arise from one of the eigenvalues being negative, reversing time will negate the Jacobian and hence the eigenvalues

Reverse Shooting

Now the original steady state is unstable and we converge to the saddle path, rather than away from it



Spectral Methods

An alternative method of solving functional equations is to use functional bases to represent them

$$f(x) = \sum_{n=0}^{\infty} z_n T_n(x)$$

The classic method of functional decomposition is the Fourier transform [$\exp(iz) = \cos(z) + i\sin(z)$]

$$\hat{f}\,(t) = \int_{-\infty}^{\infty} f(x)\exp(-2\pi i t x)dx$$

Others include Laplace, which is simply the set of polynomials

$$F(t) = \int_{0}^{\infty} f(t)\exp(-tx)dx$$

Chebyshev Polynomials

Since these are linear decompositions with known derivatives, we can easily utilize them in our recursive formulation

A common choice with value functions is the Chebyshev polynomials, which are defined by the second order difference relation

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{m+1}(x) &= 2xT_m(x) - T_{m-1}(x) \end{aligned}$$

These can be easily differentiated and hence convert a functional equation into a countable system of linear equations in ξ_n

Grid Specification

In general, we are free to choose grid points, however we want to choose grid points that render the basis orthogonal, to ensure uniqueness of the transform

$$x_k = \cos\left(\frac{2k+1}{2N}\pi\right) \quad \text{for} \quad k \in 0, \ldots, N-1$$

These satisfy the condition

$$\sum_{k=0}^{N} T_i(x_k) T_j(x_k) = \begin{cases} 0 & \text{if } i \neq j \\ N & \text{if } i = j = 0 \\ \frac{N}{2} & \text{if } i = j \neq 0 \end{cases}$$

Analytic Derivatives

We can see from the definition that the derivatives of the Chebyshev equations are

$$\begin{aligned} T_0'(x) &= 0 \\ T_1'(x) &= 1 \\ T_{m+1}'(x) &= 2xT_m'(x) + 2T_m(x) - T_{m-1}'(x) \end{aligned}$$

Substituting these into a linear differential equation will yield a linear system in z of the form $u = Qz$, which can be solved in the least squares sense (as we may have $M < N$)