Charlie Nusbaum
Econ 204B
March 7, 2018

Section 6 Notes: Introduction to Numerical Methods

Central Problem:

The central problem of computational macroeconomics is to solve the Bellman equation for an economic actor and aggregate their decisions. To do so, we need to approximate, integrate, and maximize:

$$V(x) = \underbrace{\max_y \{u(x,y) + \beta \underbrace{\mathbb{E}\, \underbrace{V(x')}_{Approx.}}_{Integ.}\}}_{Max}$$

Hence, we need to work backwards to solve these problems. Here, I will present some of the basic tools needed to complete each stead.

Numerical Differentiation:

From elementary school, we all know the definition of derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

For very small h, we can drop the limit so that

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

This formula can be thought of as a one-sided approximation to the derivative. To see this, let's take a taylor series expansion of $f(x+h)$:

$$f(x+h) \approx f(x) + h \cdot f'(x) + \frac{1}{2}h^2 f''(\xi)$$

$$\Leftrightarrow f'(x) \approx \frac{f(x+h) - f(x)}{h} + \underbrace{\frac{h}{2} f''(\xi)}_{error}$$

Thus, we are approximating the derivative at x from only a single side. To find the two-sided approximation, we must also take the taylor series expansion of $f(x-h)$:

$$f(x \pm h) \approx f(x) \pm h f'(x) + \frac{h^2}{2} \pm \frac{h^3}{6} f'''(\xi_{1,2})$$

Subtracting one from the other,

$$\Rightarrow f(x+h) - f(x-h) = 2h f'(x) + \frac{h^3}{6}(f'''(\xi_1) - f'''(\xi_2))$$

$$\Leftrightarrow f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} + \underbrace{\frac{h^2}{12}(f'''(\xi_1) - f'''(\xi_2))}_{error}$$

Notice that the 2-sided approximation takes the same number of funcion evaluations as the 1-sided approximations (2), but has a trunctation error that converges to 0 more quickly. Thus, the 2-sided approximation is unambiguously better than the 1-sided approximation.

Approximation/Interpolation:
There are many types of interpolation schemes. Some useful schemes are linear, cubic, and chebyshev interpolation. Here, I will just focus on linear interpolation. Linear interpolation simply involves creating line-segments between knots (points) so that you can evaluate your function along these line segments. I won't describe the algorithm here since it is quite simple. The properties of this interpolation scheme are given below

1) Shape preserving

2) Not twice continuously differentiable

3) Discontinous first derivatives

Root-Finding:
There are a number of root finding algorithms. I will discuss the bisection method. The bistection method finds a solution to the equation $f(x) = 0$. Note that this should look awfully similar to a rearrange Euler equation. Moreover, it is often used to solve for equilibrium prices (i.e. market clearing conditions). Bisection method requires that the function is continuous and that you supply a bracket [a,b] such that $f(a) < 0$ and $f(b) > 0$. By the intermediate value theorem we know that $\exists$ at least 1 zero in (a,b).

1) Choose a convergence criteria: $|a - b| < \epsilon$ or $|f(c)| < \delta$

2) Find $f(a)$ and $f(b)$

3) Find c such that $c = \frac{a+b}{2}$

4) If $f(c) > 0$, then $b = c$

5) If $f(c) < 0$, then $a = c$

6) Repeat 2-4 until convergence

Note that bisection method is slower than other root finders since it relies on brute force function evaluations, but it *guarentees* that you will find the solution. Gradient methods such as Newton's method can fail due to ill-behaved derivatives or approximately flat portions of your function.

Note: I don't have time for Golden-Section, but the algorithm should be clear from the code I sent you. It is basically the optimization equivalent of bisection method.

Value Function Iteration:
Here, we rely on the contraction mapping theorem so that we can repeatedly interate on our value function from any intial guess and obtain the true solution:

1) Guess $V_0(x) = 0$

2) Solve $V_1(x) = \max\{u(x, y) + \beta \cdot 0\}$

3) Solve $V_2(x) = \max\{u(x, y) + \beta \mathbb{E} V_1(x)\}$

$\vdots$

To perform the maximization step, you can either use golden-section or grid search. Grid search is perhaps even more brute force that golden-section, though it is quite robust. It entails the following:

1) Define a grid of states and choices

2) Solve for $u(x, y)$ for all possible points (outside the loop)

3) Calculate $V_0(x)$ for all possible choices

4) Choose the maximum value and associated choice

$\vdots$

Simulation:
In our models, we typically assume a unit mass of agents (infinite people). In practice, we can't do that. Instead, we just use a huge number of people and track their decisions. Thus, we require the following:

1) Policy functions

2) Many people ($\geq 10^4$)

3) A long period of time to get to steady state

A typical simulation will follow the following basic steps:

1) Initiate starting point for all agents

   - initial states
   - idiosyncratic shocks $\Rightarrow$ start at stationary distribution

2) Within a period, allow agents to make their decisions given their states using policy functions

   - Ex: choice[i,time]=interp_linear_jit(policy fxn, state)

3) Push agents' states forward

   - Ex: state'=choice[i,time]
   - Iterate Markov chain forward