INTRODUCTION TO NUMERICAL METHODS

We will learn:

• How to solve numerically a non-linear system of equations (Newton-Raphson Method)

• How to obtain numerical derivatives for an specific function

• How to solve the sequential social planner's problem

• How to implement the value function iteration method for this problem

Solving Non-Linear Equations (Newton-Raphson)

Let $x = (x_1, x_2, ..., x_n)$ be a vector of n components and $F : R^n \to R^n$

Problem: find a vector $\hat{x}$ such that $F(\hat{x}) = 0$

We denote $\bar{x}$ as the numerical approximation of the solution $\hat{x}$

Consider a first-order Taylor expansion of F around $\bar{x}$:

$$F(x) \approx F(\bar{x}) + J(\bar{x})(x - \bar{x})$$

where $J(\bar{x})$ is the Jacobian matrix of F evaluated at $\bar{x}$:

$$J(\bar{x}) = \begin{bmatrix} F_{11}(\bar{x}) & F_{12}(\bar{x}) & .... & F_{1n}(\bar{x}) \\ F_{21}(\bar{x}) & F_{22}(\bar{x}) & .... & F_{2n}(\bar{x}) \\ ..... & ..... & .... & ..... \\ F_{n1}(\bar{x}) & F_{n2}(\bar{x}) & .... & F_{nn}(\bar{x}) \end{bmatrix}$$

Using Taylor's expansion, if $\bar{x}$ is sufficiently close to the actual solution $\hat{x}$:

$$F(\hat{x}) \approx F(\bar{x}) + J(\bar{x})(\hat{x} - \bar{x})$$

then:

$$\hat{x} \approx \bar{x} - J(\bar{x})^{-1} F(\bar{x})$$

$\Longrightarrow$ this is the mathematical foundation of the Newton-Raphson method

Algorithm:

1. Propose an initial value x^0, using information of F (if possible) and initialize $s = 0$

2. Calculate the vector $F(x^s)$ and the matrix $J(x^s)$

3. Compute x^{s+1} using the following rule:

$$x^{s+1} = x^s - J(x^s)^{-1} F(x^s)$$

4. Evaluate the norm $\left\| x^{s+1} - x^s \right\|$. If the norm is greater than a pre-specified tolerance criterion, repeat step 2 with $s = s + 1$. Otherwise, finish with $\bar{x} = x^{s+1}$

There are different ways to define the norm (or distance) between two vectors of dimension n:

- Euclidean Norm:

$$\|x - y\|_2 = \left[(x_1 - y_1)^2 + ... + (x_n - y_n)^2\right]^{\frac{1}{2}}$$

- Sup-norm:

$$\|x - y\|_\infty = \max\{|x_1 - y_1|, ..., |x_n - y_n|\}$$

If $n = 1$, both norms are equivalent (and equal to the absolute value $|x - y|$)

If the initial x_0 is close enough to $\hat{x}$, we can show that Newton-Raphson method converges to $\hat{x}$

But, if x_0 initially is far from $\hat{x}$, the method could:

- Converge to a distinct solution (multiple solution case), or
- Diverge, this is, the norm $\left\|x^{s+1} - x^s\right\|$ grows at each iteration

$\Rightarrow$ We need to use different values for x_0 before arriving to a definite answer

Another disadvantage of the Newton-Raphson method is that it requires analytical expressions for all the partial derivatives of F

Numerical Derivatives

The *secant* method is similar to Newton-Raphson, but uses numerical derivatives

To do so, we write the Jacobian matrix as:

$$J(x) = [J_1(x) \quad J_2(x) \quad ... \quad J_n(x)]$$

where $J_i(x)$ is a column vector with the n partial derivatives of F with respect to x_i

Problem: find a numerical approximation for each partial derivative $J_i(x)$

Let Δ be a small number, we can approximate for each i the partial derivate "from the right"

$$J_i(x) \approx \frac{1}{\Delta}[F(x_1, ..., x_i + \Delta, ..., x_n) - F(x)]$$

or from the left:

$$J_i(x) \approx \frac{1}{\Delta}[F(x) - F(x_1, ..., \Delta - h_i, ..., x_n)]$$

It is usually recommended to average them:

$$J_i(x) \approx \frac{1}{2\Delta}[F(x_1, ..., x_i + \Delta, ..., x_n) - F(x_1, ..., x_i - \Delta, ..., x_n)]$$

unless there exists a discontinuity of F in x

The choice of Δ is arbitrary; it is recommended to use progressively smaller values, until the derivative stabilizes

`Solving the Sequential Social Planner Problem`

By solving the deterministic social planner problem, we obtain the following second order differential equation:

$$\frac{u'\left[f\left(k_t\right)-k_{t+1}+\left(1-\delta\right)k_t\right]}{\beta u'\left[f\left(k_{t+1}\right)-k_{t+2}+\left(1-\delta\right)k_{t+1}\right]}=f'\left(k_{t+1}\right)+\left(1-\delta\right)$$

that we can write in general as:

$$\Psi\left(k_t,k_{t+1},k_{t+2}\right)=0$$

Additionally we know that k_t converges monotonically to its steady state value:

$$k^*=\left(f'\right)^{-1}\left[\frac{1}{\beta}-\left(1-\delta\right)\right]$$

Problem:

Given the functional forms for u, f, and values for the parameters β y δ,

Find a sequence for k_t which solves the differential equation $\Psi\left(\bullet\right)=0$

... with initial condition $k_0>0$ and terminal condition $\lim_{t\to\infty}k_t=k^*$

Assuming that the model reaches the stationary equilibrium in a finite number of periods T, we could use directly the Newton-Raphson method

We write

$$\Psi(k_0, k_1, k_2) = 0$$

$$\Psi(k_1, k_2, k_3) = 0$$

.....

$$\Psi(k_{T-2}, k_{T-1}, k_T) = 0$$

a system of $T-1$ equations and $T+1$ unknowns

- The first missing equation is $k_0 =$ initial value
- The other missing equation could be $k_T = k^*$ or $k_T = k_{T-1}$

We can then solve this system of equations using Newton-Raphson (or the secant method)

- There are a lot of equations (at least $T = 100$), but it usually works
- We need to propose an initial sequence for $k_0^0, ..., k_T^0$

  (for example, propose the "straight line" between k_0 and $k_T = k^*$)

`Value Function Iteration`

From dynamic programming and the contraction mapping theorem, starting from *any* function v^0 (for example, $v^0(k) = 0$) the sequence v^n defined by:

$$v^{n+1}(k) = \max_{k'} \quad \left\{ u\left[f(k) + (1-\delta)k - k' \right] + \beta v^n\left(k'\right) \right\}$$

$$s.t. \quad k' \in [0, f(k) + (1-\delta)k]$$

converges to the solution v of the social planner problem, if $n \to \infty$

Lets see how to implement numerically this method to approximate the value function v

Initial configuration:

- Define a *grid* of values for k, this is, a vector:

  $$K = \left(K_1, K_2, ..., K_p\right)$$

  with $K_1 = K_{\min}$ and $K_p = K_{\max}$

  For simplicity, we could use equally spaced points:

  $$K_2 = K_{\min} + \eta, \quad K_3 = K_{\min} + 2\eta, \qquad etc.$$

  with $\eta = (K_{\max} - K_{\min}) / (p-1)$

  If the size of the grid p increases, the approximation is more precise, but the algorithm slows down

- Define the matrix M as:

$$M = \begin{bmatrix} F(K_1, K_1) & F(K_1, K_2) & ..... & F(K_1, K_p) \\ F(K_2, K_1) & F(K_2, K_2) & ..... & F(K_2, K_p) \\ ..... & ..... & ..... & ..... \\ F(K_p, K_1) & F(K_p, K_2) & ..... & F(K_p, K_p) \end{bmatrix}$$

with $F(x, y) = u[f(x) + (1 - \delta)x - y]$

M stores the return function evaluated in each possible combination (k, k') of the grid

- Eliminate the cells that are not feasible by setting:

$$M_{ij} \equiv F\left(K_i, K_j\right) = -10000 \qquad \text{si} \quad K_j > f(K_i) + (1 - \delta)K_i$$

Algorithm:

1. Propose an initial column vector $V^0 \in R^p$ and initialize $s = 0$ (for example, $V^0 = 0$)

2. Given V^s and M, calculate V_i^{s+1} for all $i \in \{1, .., p\}$ as:

$$V_i^{s+1} = \max_{j \in \{1,..,p\}} \left\{ M_{i,j} + \beta V_j^s \right\}$$

3. Calculate $\left\| V^{s+1} - V^s \right\|$. If the norm is greater than the previously specified tolerance criterion, return to step 2 with $s = s + 1$. Otherwise, the algorithm finishes with $V = V^{s+1}$

The result will be an approximation of the value function v at each point of the grid:

$$V = \begin{bmatrix} V(K_1) \\ V(K_2) \\ ... \\ V(K_p) \end{bmatrix} \approx \begin{bmatrix} v(K_1) \\ v(K_2) \\ ... \\ v(K_p) \end{bmatrix}$$

The algorithm also stores the decision rule G, where G_i for all $i \in \{1, .., p\}$ is given by:

$$G_i = \underset{j \in \{1,..,p\}}{\arg\max} \left\{ M_{i,j} + \beta V_j^s \right\}$$

which allow us, starting off any $k_0 = K_i$, to recover the optimal sequence for the capital

Solving the Recursive Competitive Equilibrium

The iteration method of the value function is not particularly suitable for directly solving the recursive competitive equilibrium, since it requires:

- Two state variables (individual and aggregate capital)$\Leftarrow$ not important
- Law of motion of aggregate capital Γ is an unknown equilibrium object when the consumer solves his Bellman equation

We need to follow a double iteration algorithm

Assume that the law of motion follows a polynomial of degree n:

$$K' = \Gamma(K) = a_0 + a_1 K + a_2 K^2 + ... + a_n K^n$$

Algorithm:

1. Propose an initial parameter vector $(a_0, a_1, ..., a_n)$

2. Given Γ, solve the consumer's Bellman equation by iterating the value function and obtain the optimal sequence $k_0, k_{1,}..., k_T$

3. Using time series $k_0, k_{1,}..., k_T$, run the following regression:

   $$k_{t+1} = a_0 + a_1 k_t + a_2 k_t^2 + ... + a_n k_t^n$$

   and estimate a new vector of parameters $(\hat{a}_0, \hat{a}_1, ..., \hat{a}_n)$

4. Compare $(\hat{a}_0, \hat{a}_1, ..., \hat{a}_n)$ and $(a_0, a_1, ..., a_n)$. If the distance is greater than the tolerance criterion, go back to step 2 with the updated law of motion. Otherwise, the algorithm converges

This method will be more accurate the higher the degree n of the polynomial

Even with large n, the convergence is not guaranteed