Nested Methods and a Superior Approach

October 17, 2010

Nested Methods are Common

- General equilibrium
  - Problem is
$$0 = E(p)$$
  - Any method requires computation of $E(p)$, but $E(p)$ often cannot be expressed analytically. Hence, people use numerical methods to compute $E(p)$
    - Suppose you use a fixed point method, such as in
$$p_{k+1} = D^{-1}(S(p_k))$$
    - Inner loop: Each iteration requires the numerical computation of S (probably an optimization problem for firms given prices) and D^{-1} (just computing the marginal utilities); these computations will often involve iteration until some stopping rule is satisfied
    - Outer loop: Iteration over prices continues until stopping rule is satisfied
    - What should the stopping rules be?
  - General principle: Any optimization or nonlinear equation solver will reduce precision by roughly half; that is, if the inputs are accurate up q digits, the outputs can be accurate only up to $q/2$ digits and the stopping rule cannot demand better.
  - General principle for nested methods: If the stopping rule for the inner loop demands q digit accuracy, then the stopping rule for the outer loop can demand only $q/2$ digits.

- Maximization: Suppose that we want to solve

$$\max f\left(x, Y\left(x\right)\right)$$

where $Y\left(x\right)$ is the solution to some other numerical problem described by $g\left(x, y\right) = 0$. Nested approach has two layers

  - $*$ Inner loop: compute $Y\left(x\right)$: standard practice is to write code to solve this problem - BAD idea!!!!
  - $*$ Outer loop: for each x, compute $f\left(x, Y\left(x\right)\right)$ in an unconstrained optimization algorithm, again, usually with user-written code - BAD idea!!!

- General experience: Nested methods are slow and prone to being inaccurate
  - If you use a slow method for inner loop, you will tend to set a loose stopping rule
    - $*$ The loose inner stopping rule will often lead to nonconvergence for outer loop
    - $*$ In order to get convergence, you will need to set a very loose stopping rule for outer loop
    - $*$ Result will be bad.
  - Even if you use good algorithms, you will need to compute $Y\left(x\right)$ for each value of x used in the outer loop
    - $*$ Finite difference methods are often the only way you can take derivatives in outer loop
    - $*$ The slowness will lead you to do inferior econometrics: no bootstrapping, avoid full information estimators

Optimization to the Rescue!

- Suppose that you want to solve

$$\max f\left(x, Y\left(x\right)\right)$$

  where $Y\left(x\right)$ is the solution to some other numerical problem.

$$0 = g\left(x, y\right)$$

  - Reformulate problem as

$$\begin{aligned} \max_{x,y} \quad & f\left(x, y\right) \\ \text{s.t. } 0 = & g\left(x, y\right) \end{aligned}$$

  - Advantages
    - This can be sent to a constrained optimization solver written by professionals; professionals do not write NFXP code.
    - No need for you to construct an algorithm to compute $Y\left(x\right)$
    - You can set tight stopping rules for all variables, y and x.
    - You can try several algorithms to find the one that works best
  - Disadvantages
    - Too large IF you don't use good solvers that (a) exploit sparseness, and (b) use automatic differentiation.
    - Uses substantial amount of memory, but that is not a problem today.

- Lesson: Learn some math so that you can get the computer to do the hard work