Nested Fixed-Point versus MPEC Methods

March 12, 2020

Nested Methods are Common: GE example

- General equilibrium problem is

$$0 = E(p)$$

  where $E(p)$ cannot be expressed analytically.
- 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)
  - 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.

Estimation: Simple Consumer Demand Example

- Data and Model
  - Data on demand, q, and price p, but demand is observed with error ε.
  - True demand is $q - \varepsilon$.
  - Assume a parametric form for utility function $u(c; \beta)$ where β is a vector of parameters.
  - Economic theory implies

$$u_c(c; \beta) = u_c(q - \varepsilon; \beta) = p$$

- Standard Approach (from Econ 712, University of Wisconsin, 1979)
  - Assume, for example, a functional form for utility

$$u(c) = c - \beta c^2.$$

  - Solve for demand function

$$c = (1 - p)/(2\beta)$$

  - Hence, i'th data point satisfies

$$q_i = (1 - p_i)/(2\beta) + \varepsilon_i$$

    for some ε_i.
  - To estimate β, choose β to minimize the sum of squared errors

$$\sum_{i=1} (q_i - (1 - p_i)/(2\beta))^2 .$$

- Limitations
  - Need to solver for demand function, which is hard if not impossible
  - For example, suppose

    $$u(c) = c - \beta\left(c^2 + c^4 + c^6\right)$$

    with first-order condition

    $$1 - \beta\left(2c + 4c^3 + 6c^5\right) = p$$

  - There is no closed-form solution for demand function.
  - What were you taught to do in this case? *Change the model*!

- MPEC Procedure
  - MPEC is Mathematical Programming with Equilibrium Conditions
  - Deal with the first-order condition directly since it has all the information you can have.
  - Recognize that all you do is find the errors that minimize their sum of squares but are consistent with structure

- Quadratic utility function example
  - For our consumption demand model, this is the problem

$$\begin{aligned} \min_{\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & u_c\left(q_i - \varepsilon_i; \beta\right) = p_i \end{aligned}$$

  - In the case of the quadratic utility function, this reduces to

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & 1 - 2\beta c_i = p_i \\ & c_i = q_i - \varepsilon_i \end{aligned}$$

- Degree-six utility function
  - This reduces to the problem

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & 1-\beta\left(2c_i+4c_i^3+6c_i^5\right)=p_i \\ & c_i = q_i - \varepsilon_i \end{aligned}$$

  - You cannot solve out the ε's but you can still do least squares estimation

- Even when you can solve for demand function, you may not want to.
  - Consider the case

$$
\begin{aligned}
u(c) &= c - \beta_1 c^2 - \beta_2 c^3 - \beta_3 c^4 \\
u'(c) &= 1 - 2\beta_1 c - 3\beta_2 c^2 - 4\beta_3 c^3
\end{aligned}
$$

  - Demand function is

$$
\begin{aligned}
q &= \frac{1}{12\beta_3} W - \frac{1}{4} \frac{8\beta_1\beta_3 - 3\beta_2^2}{\beta_3 W} - \frac{1}{4}\frac{\beta_2}{\beta_3} \\
W &= \sqrt[3]{\left(108\beta_1\beta_2\beta_3 - 216\beta_3^2 p + 216\beta_3^2 - 27\beta_2^3 + 12\sqrt{3}\beta_3 Z\right)} \\
Z &= \sqrt{Z_1 + Z_2} \\
Z_1 &= 32\beta_1^3\beta_3 - 9\beta_1^2\beta_2^2 - 108\beta_1\beta_2\beta_3 p + 108\beta_1\beta_2\beta_3 \\
Z_2 &= 108\beta_3^2 p^2 - 216\beta_3^2 p + 27 p\beta_2^3 + 108\beta_3^2 - 27\beta_2^3
\end{aligned}
$$

  - Demand function is far costlier to compute than the first-order conditions.
- The (*bad*) habit of restricting models to cases with closed-form solutions is completely unnecessary.

Nested Fixed-point Iteration

Suppose Z is a collection of exogenous numbers, and that we want to solve

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

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

- Example: Assume
  - Z is the data,
  - x is a vector of parameters,
  - $Y\left(x\right)$ expresses economic functions (supply, demand, investment,...) if x were true, and
  - $f\left(x, Y\left(x; Z\right)\right)$ is the likelihood of observing Z if x were the true parameter values.
- Nested approach has two layers
  - Inner loop: compute $Y\left(x\right)$: standard practice is to write amateur code to solve this problem - BAD idea!!!!
  - Outer loop: for each x, compute $f\left(x, Y\left(x\right), Z\right)$ in an unconstrained optimization algorithm, again, usually with user-written code - BAD idea!!!

- General experience: Nested methods are slow and 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 $Y(x)$ 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
    - If for some x there are multiple solutions for $g(x, y) = 0$, you must compute ALL of them!

Constrained Optimization to the Rescue!

- Suppose that you want to solve (drop Z from the notation)

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

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

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

- MPEC approach is to reformulate problem as

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

- Advantages
  - Can use a solver written by professionals
    - Professionals do not write NFXP code
    - They use the term "implicit programming" to describe NFXP
  - 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: Problem is too large IF you don't
  - use good solvers
  - exploit sparseness
  - use automatic differentiation.
- Memory requirements are less with NFXP
  - Rust's NFXP was the best way to go in 1986 when you could have only a small amount of RAM
  - Memory is not a problem today, 34 years later.
- Questions
  - Do you listen to the music your parents liked?
  - Do you wear the clothes your parents liked?
- Lesson: Learn some math so that you can get the computer to do the hard work