Analyzing Environmental Policies with IGEM, an Intertemporal General Equilibrium Model of U.S. Growth and the Environment

Part 2

Appendix I. Model Solution Algorithm

Dale Jorgenson Associates
Cambridge, MA

April 2008

Appendix I. Model Solution Algorithm

IGEM is solved using a suite of nested solution algorithms. This appendix provides an overview of the solution process and discusses some of the key details of the implementation.

I.1 Structure of the Algorithm

Conceptually, the model consists of three main components: (1) an intratemporal module; (2) a steady-state module; and (3) an intertemporal module. The intratemporal module computes a complete equilibrium for any given year *t* conditional on that year's exogenous variables and the values of two intertemporal variables: *k* and *full_con* (the capital stock and the value of full consumption). The steady-state module, in contrast, iterates over values of *k* and *full_con* and calls the intratemporal module repeatedly until it finds a (*k*,*full_con*) pair satisfying the model's steady state conditions. Finally, the intertemporal module that iterates over complete intertemporal trajectories of *k* and *full_con* (invoking the intratemporal module for every period) until it finds a set that satisfy the model's accumulation and Euler conditions. In addition, the intertemporal module ensures that the trajectories satisfy two boundary conditions: (1) the initial capital stock matches the value of the capital stock in the model's data set, and (2) the value of full consumption in the final period of the simulation matches its steady state value.

In terms of practical implementation, the algorithm begins by using the enhanced version of Newton's Method described in section I.2 to compute the model's steady state capital stock and value of full consumption (*k* and *full_con*). The algorithm itself is

implemented in subroutine *newtonss* and the Newton miss distances are computed by *ss_miss*. At each iteration of the steady state module (that is, for each trial pair of *k* and *full_con*), the algorithm invokes the intratemporal module (which will be described in more detail below) to compute all other steady state variables conditional on the guess of *k* and *full_con*. After the steady state has been obtained, the intertemporal algorithm begins iterating over the trajectories of *k* and *full_con* using the hybrid intertemporal algorithm described in section I.3. The algorithm is implemented in subroutine *ftaylor* and subroutine *path* is called at each iteration to evaluate each trajectory. When the intertemporal algorithm converges, the simulation program saves the final transition path for *full_con* (which is formally the model's costate variable), and then writes out a complete set of results for all endogenous variables.

During both the steady state calculation and intertemporal calculations, the intratemporal module is invoked repeatedly. For computational efficiency, it consists of a three-tiered suite of Newton's Method algorithms (again, enhanced as described in I.2). The outer tier is implemented in subroutine *newtonfp* and iterates over a vector of factor prices and other variables. The corresponding miss distances are computed by subroutine *intra_miss*. For each iteration of *newtonfp*, an inner algorithm is called to determine industry output prices conditional on the guessed vector of factor prices. The inner algorithm is implemented in *newtonpo* and the miss distances are calculated by *po_miss*. Finally, for each guess of factor prices a second inner algorithm computes the aggregate price of consumption. The algorithm is implemented in *newtonpcc* with the miss distance calculated in *pcc_miss*. Using a nested structure allows the algorithm to find a solution

far more quickly than it would if the complete set of intratemporal equations were solved in a single tier.

I.2 Broyden's Modification to Newton's Method

Broyden (1965, 1973) has developed a procedure that can reduce the number of function evaluations required to solve a system of equations using Newton's method. This section describes how the method works.

Newton's method is an iterative procedure used to find a vector *x* that satisfies an equation *f*(*x*) = 0 where *f* is a vector-valued function. Roughly speaking, a guess of *x* is refined repeatedly until the each element of *f* is approximately zero. The fundamental relationship used to improve the guess of *x* can be derived from a first-order Taylor series expansion of *f* about a trial solution vector *x*. Suppose that the value of *x* at iteration *k* is x_k, and that evaluating *f* at x_k gives f_k. If the Jacobian of *f* at x_k is J_k, the Newton adjustment s_k and the new trial solution x_{k+1} are given by the equations below:

$$s_k = -J_k^{-1} f_k \tag{I.1}$$

$$x_{k+1} = x_k + s_k \tag{I.2}$$

In practice, Newton's method is usually implemented as follows. Given a trial solution x_k, the value of f_k is computed. If f_k is not sufficiently close to zero (usually determined by computing $f_k' f_k$), J_k is formed by perturbing each of the n elements of x_k in succession. J_k is then used to determine s_k using equation (1), and the new trial solution, x_{k+1}, is found from equation (2). For each iteration f must be evaluated $n+1$ times—once to obtain f_k and n times to produce J_k.

Broyden's modification is a particular way of using f_{k+1} to form J_{k+1} from J_k without additional function evaluations. At each iteration the Jacobian will be less accurate than under the conventional algorithm so more iterations are usually required for convergence. Even so, the computational gain may still be substantial since the number of function evaluations per iteration is reduced from $n+1$ to 1.

The Jacobian updating procedure works as follows. Let $y_k = f_{k+1} - f_k$. Since $y_k / |s_k|$ is the directional derivative of f in the direction given by s_k, y_k can be used to revise the Jacobian. It does not, however, determine a unique adjustment to J_k, so Broyden imposes the additional conditions that the directional derivatives implied by J_k in directions orthogonal to s_k be preserved in J_{k+1}. This produces the updating rule given below:

$$J_{k+1} = J_k + (y_k - J_k s_k)\frac{s_k'}{s_k' s_k} \qquad \text{(I.3)}$$

What is really needed for Newton's method, however, is J_k^{-1}. Broyden has also derived an updating formula which operates directly on the inverse, eliminating numerical difficulties which would arise from constantly inverting J. This update is given below:

$$J_{k+1}^{-1} = J_k^{-1} + (s_k - J_k^{-1} y_k)\frac{s_k' J_k^{-1}}{s_k' J_k^{-1} y_k} \qquad \text{(I.4)}$$

For the updating method to work, an initial value of J is required. In most cases, this must be constructed by the usual method of perturbing x n times.[1] When a number of similar problems are to be solved, however, the final Jacobian from the previous problem is often a good approximation to the true Jacobian for the next problem. In this case, using the old Jacobian as an initial guess eliminates the *n* function evaluations at the beginning of the new problem. This results in a substantial increase in the speed of the algorithm and is implemented in the model.

I.3 A Hybrid Intertemporal Algorithm

This algorithm is a generalization of the method due to Fair and Taylor (1983), and exploits the fact that most economic models contain an accumulation equation relating state variables in adjacent periods. It is substantially faster than the ordinary Fair-Taylor algorithm while providing equally accurate results. The method is termed "hybrid" because it employs certain features of multiple shooting (see Lipton, *et al*., 1982) obtain these improvements in performance. In a sense, multiple shooting and the Fair-Taylor approach are at different ends of a single spectrum. Shooting uses relatively few intermediate points, but employs a great deal of information about the problem's dynamic features. Fair-Taylor, on the other hand, uses a large number of points and almost no dynamic information. The method described here lies in between because it uses many points, but also a certain amount of dynamic data.

Given a vector valued function A that generates a set of actual values from a vector of expectations E, the problem is to find a vector E which solves the equation:

---

[1] In principle, any matrix could be used as an initial Jacobian. However, convergence will usually be

$$E = A(E) \tag{I.5}$$

The Fair-Taylor algorithm proceeds by computing $A(E_k)$ for a trial solution E_k. If A_k is not sufficiently close to E_k, a new trial vector is determined as shown:

$$E_{k+1} = \gamma A_k + (1-\gamma)E_k \tag{I.6}$$

where $A_k = A(E_k)$, and γ is a parameter used to ensure stability. An important feature of this approach is that the revision of a particular element of E depends only on the corresponding elements of A_k and E_k: no information about adjacent elements is used. In many economic problems, however, E^i and E^{i+1} are related by an accumulation equation. Furthermore, steady state values of E are usually known. Together, these features mean that extending the algorithm to employ information about adjacent periods could lead to a substantial improvement in convergence speed.

An alternative technique can be developed by replacing *A* with a first-order Taylor series expansion as shown:

$$A(E_{k+1}) \approx A(E_k) + J_k(E_{k+1} - E_k) \tag{I.7}$$

Taking E_{k+1} to be a solution, the left hand side can be replaced to give the following:

$$E_{k+1} = A_k + J_k(E_{k+1} - E_k) \tag{I.8}$$

Collecting unknown terms on the left yields the equation below:

$$(I - J_k)E_{k+1} = A_k - J_k E_k \tag{I.9}$$

For convenience define vector Φ_k and rewrite the equation as shown below:

$$\Phi_k = A_k - J_k E_k \tag{I.10}$$

---

slower if it isn't reasonably close to the truth.

$$(I - J_k)E_{k+1} \quad = \quad \Phi_k \tag{I.11}$$

Writing out a small problem in scalar form (where the subscript *k* on ***J*** has been eliminated for clarity):

$$\begin{bmatrix} 1-J_{11} & -J_{12} & -J_{13} \\ -J_{21} & 1-J_{22} & -J_{23} \\ -J_{31} & -J_{32} & 1-J_{33} \end{bmatrix} \begin{bmatrix} E^1_{k+1} \\ E^2_{k+1} \\ E^3_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi^1_k \\ \Phi^2_k \\ \Phi^3_k \end{bmatrix} \tag{I.12}$$

In practice the true array $I - J_k$ will not be available, and a numerical approximation will have to be used. Often the problem can be set up so that the actuals in period *i* depend on expectations no farther in the future than period *i*+1. In the example above, this means that J_{13} will be zero. Further simplification can be achieved by setting the derivatives of the actuals with respect to past expectations to zero. (Note that unlike the previous tactic this is an approximation, since these terms will usually be small but nonzero.) One final approximation is to assume that the remaining partials are the same across periods. This produces the system below:

$$\begin{bmatrix} 1-J_{11} & -J_{12} & 0 \\ 0 & 1-J_{11} & -J_{12} \\ 0 & 0 & 1-J_{11} \end{bmatrix} \begin{bmatrix} E^1_{k+1} \\ E^2_{k+1} \\ E^3_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi^1_k \\ \Phi^2_k \\ \Phi^3_k \end{bmatrix} \tag{I.13}$$

Finally, expanding the right hand side gives:

$$\begin{bmatrix} 1-J_{11} & -J_{12} & 0 \\ 0 & 1-J_{11} & -J_{12} \\ 0 & 0 & 1-J_{11} \end{bmatrix} \begin{bmatrix} E^1_{k+1} \\ E^2_{k+1} \\ E^3_{k+1} \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} - \begin{bmatrix} J_{11} & J_{12} & 0 \\ 0 & J_{11} & J_{12} \\ 0 & 0 & J_{11} \end{bmatrix} \begin{bmatrix} E^1_k \\ E^2_k \\ E^3_k \end{bmatrix} \tag{I.14}$$

The Fair-Taylor method is equivalent to setting J_{11} and J_{12} to zero. The algorithm can be improved if values of these partials are available, even if they must be found numerically. If the steady state value of *E* is known, the equation for the last actual can

be dropped and the final element of *E* set directly to the steady state. This helps determine earlier values of *E*, since the periods are linked by the J_{12} terms in the *I-J* matrix.

For a model which has one foresight variable, the above system of equations can be solved easily by backward substitution; it is not necessary to use Gaussian elimination or matrix inversion. The new expectation for the last period of the problem above can be found as shown:

$$E^3_{k+1} = \frac{1}{1-J_{11}}\left(A_3 - J_{11}E^3_k\right) \tag{I.15}$$

Earlier periods are found by repeated application of the equation:

$$E^i_{k+1} = \frac{1}{1-J_{11}}\left(A_i - J_{11}E^i_k - J_{12}E^{i+1}_k + J_{12}E^{i+1}_{k+1}\right) \tag{I.16}$$

For two or more variables, the method is slightly more complicated because it requires the inverse of matrix (I - J_{11}). A typical revised expectation is calculated as shown:

$$E^i_{k+\mathbf{1}} = \left(I - J_{\mathbf{11}}\right)^{-1}\left(A_i - J_{\mathbf{11}}E^i_k - J_{\mathbf{12}}E^{i+\mathbf{1}}_k + J_{\mathbf{12}}E^{i+\mathbf{1}}_{k+\mathbf{1}}\right) \tag{I.17}$$

Since $J_{\mathbf{11}}$ is assumed to be constant over periods, it is only necessary to compute $\left(I - J_{\mathbf{11}}\right)^{-1}$ once for each iteration of the algorithm.