THE EFFECTS OF ENVIRONMENTAL REGULATION AND ENERGY PRICES ON U.S. ECONOMIC PERFORMANCE

A thesis presented

by

Peter Jensen Wilcoxen

to

The Department of Economics

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Economics

Harvard University

Cambridge, Massachusetts

December 1988

Appendix E

E. NUMERICAL METHODS

This appendix describes three unusual numerical techniques that were of central importance to the model. The first, Broyden's Method is a modification to Newton's Method that is much faster for problems in which computing the jacobian of the residual functions is difficult or costly. The second method is generalization of the Fair-Taylor approach for solving rational expectations models. Its design allows it to take advantage of a common feature of such models to substantially reduce the computer time required to obtain a perfect foresight solution. The final section describes a method of adjusting input-output tables to obtain consistent row and column sums. It is originally due to Kuroda, and has significant advantages over the traditional RAS method.

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

$$\mathbf{s}_k = -\mathbf{J}_k^{-1}\mathbf{f}_k \tag{E.1}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k \tag{E.2}$$

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

Broyden's modification is a particular way of using $\mathbf{f}_{k+1}$ to form $\mathbf{J}_{k+1}$ from $\mathbf{J}_k$ 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 $\mathbf{y}_k=\mathbf{f}_{k+1}-\mathbf{f}_k$. Since $\mathbf{y}_k/|\mathbf{s}_k|$ is the directional derivative of $\mathbf{f}$ in the direction given by $\mathbf{s}_k$, $\mathbf{y}_k$ can be used to revise the Jacobian. It does not, however, determine a unique adjustment to $\mathbf{J}_k$, so Broyden imposes the additional conditions that the directional derivatives implied by $\mathbf{J}_k$ in directions orthogonal to $\mathbf{s}_k$ be preserved in $\mathbf{J}_{k+1}$. This produces the updating rule given below:

$$\mathbf{J}_{k+1} = \mathbf{J}_k + (\mathbf{y}_k - \mathbf{J}_k\mathbf{s}_k)\frac{\mathbf{s}_k'}{\mathbf{s}_k'\mathbf{s}_k} \tag{E.3}$$

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

$$\mathbf{J}_{k+1}^{-1} = \mathbf{J}_k^{-1} + (\mathbf{s}_k - \mathbf{J}_k^{-1}\mathbf{y}_k)\frac{\mathbf{s}_k'\mathbf{J}_k^{-1}}{\mathbf{s}_k'\mathbf{J}_k^{-1}\mathbf{y}_k} \tag{E.4}$$

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 can result in a substantial increase in the speed of the algorithm.

Broyden's method is only one of a number of updating procedures available; some other procedures are discussed in Press, *et al.* (1986). The use of any updating procedure could result in substantial savings, particularly when combined with reuse of previous Jacobians. In one application the number of function calls was reduced by 37% when updating alone was used; adding strategic reuse of old Jacobians brought the number of function evaluations down to 26% of the original number.

E.2. 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.

---

1. In principle, any matrix could be used as an initial jacobian. However, convergence will usually be slower if it isn't reasonably close to the truth.

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:

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

The Fair-Taylor algorithm proceeds by computing $\mathbf{A}(\mathbf{E}_k)$ for a trial solution $\mathbf{E}_k$. If $\mathbf{A}_k$ is not sufficiently close to $\mathbf{E}_k$, a new trial vector is determined as shown:

$$\mathbf{E}_{k+1} = \gamma\mathbf{A}_k + (1-\gamma)\mathbf{E}_k \tag{E.6}$$

where $\mathbf{A}_k = \mathbf{A}(\mathbf{E}_k)$, 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 $\mathbf{A}_k$ and $\mathbf{E}_k$ – 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:

$$\mathbf{A}(\mathbf{E}_{k+1}) \approx \mathbf{A}(\mathbf{E}_k) + \mathbf{J}_k(\mathbf{E}_{k+1}-\mathbf{E}_k) \tag{E.7}$$

Taking $\mathbf{E}_{k+1}$ to be a solution, the left hand side can be replaced to give the following:

$$\mathbf{E}_{k+1} = \mathbf{A}_k + \mathbf{J}_k(\mathbf{E}_{k+1}-\mathbf{E}_k) \tag{E.8}$$

Collecting unknown terms on the left yields the equation below:

$$(\mathbf{I}-\mathbf{J}_k)\mathbf{E}_{k+1} = \mathbf{A}_k - \mathbf{J}_k\mathbf{E}_k \tag{E.9}$$

For convenience define vector $\mathbf{\Phi}_k$ and rewrite the equation as shown below:

$$\mathbf{\Phi}_k = \mathbf{A}_k - \mathbf{J}_k\mathbf{E}_k \tag{E.10}$$

$$(\mathbf{I}-\mathbf{J}_k)\mathbf{E}_{k+1} = \mathbf{\Phi}_k \tag{E.11}$$

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

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

In practice the true array $\mathbf{I}-\mathbf{J}_k$ 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:

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

Finally, expanding the right hand side gives:

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

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:

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

Earlier periods are found by repeated application of the equation:

$$E_{k+1}^i = \frac{1}{1-J_{11}} \left(A_i - J_{11} E_k^i - J_{12} E_k^{i+1} + J_{12} E_{k+1}^{i+1} \right) \tag{E.16}$$

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

$$\mathbf{E}_{k+1}^{i} = (\mathbf{I}-\mathbf{J}_{11})^{-1}\left(\mathbf{A}_i - \mathbf{J}_{11}\mathbf{E}_k^i - \mathbf{J}_{12}\mathbf{E}_k^{i+1} + \mathbf{J}_{12}\mathbf{E}_{k+1}^{i+1} \right) \tag{E.17}$$

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

E.3. Kuroda's Method for Constructing Consistent Input-Output Data Sets

This section describes the method used to resolve inconsistencies between input-output data from various sources. The approach used was originally proposed by Kuroda (1988), and is superior to the RAS method of Stone in several respects.

In input-output analysis it is often necessary to use inconsistent data sets originating from different government agencies. For example, the table of interindustry transactions created by one agency may not be consistent with value-added and final demand vectors produced elsewhere. In this case, the investigator will be confronted with three pieces of data which do not agree: a table of interindustry transactions, a vector of commodity outputs, and a vector of gross outputs by industry. The task then becomes adjusting the transactions table to match the commodity and industry output vectors.

In the past, this problem has been solved by using the RAS method. RAS is an iterative algorithm which scales the rows and columns of the transactions table up and down repeatedly until the table's row and column sums agree with the target vectors. It has been shown that RAS will eventually converge, but the result will not necessarily be close in any economic sense to the original transactions table. The purpose of this paper is to define a measure of how far a new transactions table is from the original, and to derive an algorithm which will

construct a table minimizing that distance.

Given an $n \times m$ matrix X^o of initial data, define r_{ij} and c_{ij} to be the shares of each element in the row and column sums of the original matrix:

$$r_{ij} = \frac{X_{ij}^o}{\sum_{j=1}^{m} X_{ij}^o}, \qquad c_{ij} = \frac{X_{ij}^o}{\sum_{i=1}^{n} X_{ij}^o} \tag{E.18}$$

Let R be a vector of target row totals, and C a vector of desired column totals. The following function can then be used to measure the distance between a revised matrix X and the original (embodied in r and c), where w and v are arbitrary sets of weights:

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}}{R_i} - r_{ij}\right)^2 w_{ij} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}}{C_j} - c_{ij}\right)^2 v_{ij} \tag{E.19}$$

It is now possible to choose X to minimize this function subject to the following constraints:

$$R_i = \sum_{j=1}^{m} X_{ij} \tag{E.20}$$

$$C_j = \sum_{i=1}^{n} X_{ij} \tag{E.21}$$

The Lagrangian for this problem is:

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}}{R_i} - r_{ij}\right)^2 w_{ij} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}}{C_j} - c_{ij}\right)^2 v_{ij}$$
$$+ \sum_{i=1}^{n}\lambda_i\left(R_i - \sum_{j=1}^{m} X_{ij}\right) + \sum_{j=1}^{m}\mu_j\left(C_j - \sum_{i=1}^{n} X_{ij}\right) \tag{E.22}$$

Taking first order conditions gives:

$$\frac{\partial L}{\partial X_{ij}} = \left(\frac{X_{ij}}{R_i} - r_{ij} \right) \frac{w_{ij}}{R_i} + \left(\frac{X_{ij}}{C_i} - c_{ij} \right) \frac{v_{ij}}{C_i} - \lambda_i - \mu_j = 0 \tag{E.23}$$

Collect terms in X_{ij}, and for convenience make the following definitions:

$$S_{ij} = \left(\frac{w_{ij}}{R_i^2} + \frac{v_{ij}}{C_j^2} \right)^{-1} \tag{E.24}$$

$$G_{ij} = \frac{r_{ij} w_{ij}}{R_i} + \frac{c_{ij} v_{ij}}{C_j} \tag{E.25}$$

This allows the first order conditions to be rewritten as shown

$$X_{ij} = S_{ij}(\lambda_i + \mu_j + G_{ij}) \tag{E.26}$$

Both S and G depend only on initial data and the weights w and v, so it is only necessary to determine λ and μ, to find the optimal X_{ij}. These may be determined by applying the constraints:

$$R_i = \sum_{j=1}^{m} X_{ij} = \sum_{j=1}^{m} S_{ij}(\lambda_i + \mu_j + G_{ij}) \tag{E.27}$$

$$C_j = \sum_{i=1}^{n} X_{ij} = \sum_{i=1}^{n} S_{ij}(\lambda_i + \mu_j + G_{ij}) \tag{E.28}$$

Define diagonal matricies S^R and S^C as indicated:

$$S^R_{ii} = \sum_{j=1}^{m} S_{ij}, \qquad S^C_{jj} = \sum_{i=1}^{n} S_{ij} \tag{E.29}$$

In matrix notation, the constraints can now be expressed as

$$S^R \cdot \lambda + S \cdot \mu = R - A \tag{E.30}$$

$$S' \cdot \lambda + S^C \cdot \mu = C - B \tag{E.31}$$

where A and B are vectors defined as follows:

$$A_i = \sum_{j=1}^{m} S_{ij} G_{ij}, \qquad B_i = \sum_{i=1}^{n} S_{ij} G_{ij} \tag{E.32}$$

With more manipulation, it is possible to derive explicit formulae for λ and μ. For computational purposes, however, it is better to arrange the equations into the following system, which can be solved easily by any competent numerical package:

$$\begin{bmatrix} S^R & S \\ S' & S^C \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} R-A \\ C-B \end{bmatrix} \tag{E.33}$$

Armed with the values of λ and μ, the optimal choice of X_{ij} can be computed directly.

It is worthwhile to examine a few of the possible weighting schemes that can be used. The most obvious approach is to weight all errors equally, which means that $w_{ij}=v_{ij}=1$ for all

i and j. On the other hand, the following choice of weights results in a drastic simplification of the revision formula:

$$w_{ij} = \frac{1}{2}R_i^2, \qquad v_{ij} = \frac{1}{2}C_j^2 \tag{E.34}$$

This means that S_{ij}=1 for all i and j. Moreover, the following is true of G_{ij}:

$$G_{ij} = \frac{1}{2}(r_{ij}R_i + c_{ij}C_j) \tag{E.35}$$

which is simply the average of the values obtained by applying the original shares to the target row and column sums. Furthermore, it can be shown that all of the following are true:

$$\sum_{i=1}^{n}\lambda_i = 0, \qquad \sum_{j=1}^{m}\mu_j = 0, \tag{E.36}$$

$$S_{ij}^R = m, \qquad S_{ij}^C = n. \tag{E.37}$$

This means that the revision formula has a particularly simple form:

$$X_{ij} = G_{ij} + \frac{1}{m}(R_i - \sum_{j=1}^{m}G_{ij}) + \frac{1}{n}(C_j - \sum_{i=1}^{n}G_{ij}) \tag{E.38}$$

Thus, the revised X_{ij} is just G_{ij} (which has the interpretation above) adjusted to correct the row and column sums. It is important to note that this method does not guarantee that all elements of X will be nonnegative.

Kuroda proposes a different weighting scheme, with w and v determined by the following equations:

$$w_{ij} = \frac{1}{r_{ij}^2}, \qquad v_{ij} = \frac{1}{c_{ij}^2}. \tag{E.39}$$

This choice of weights causes Q to be a function of the percentage changes in the coefficients:

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}/R_i}{r_{ij}} - 1\right)^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{X_{ij}/C_j}{c_{ij}} - 1\right)^2 \tag{E.40}$$

In most cases, this will ensure that all elements of X are positive, since making one negative would require a change of more than one hundred percent, resulting in a large value of Q. It is possible, however, for negative numbers to arise if the row and column targets differ substantially from the corresponding totals of the initial array.