RESUELVE: A *GAUSS* PROGRAM FOR SOLVING COMPUTABLE GENERAL EQUILIBRIUM AND DISEQUILIBRIUM MODELS

Carlos M. Urzúa
El Colegio de México

Resumen: Este artículo contiene un programa escrito en *GAUSS* que puede usarse, entre otras cosas, para encontrar el equilibrio en modelos computables de equilibrio general, y el equilibrio con rigideces de precios en modelos no-walrasianos. También se proveen aplicaciones simples a esos dos casos, junto con aplicaciones a programación lineal y cuadrática.

Abstract: This paper describes a *GAUSS* program that can be used, among other things, to find equilibria for computable general equilibrium models, and fix-price equilibria for general non-Walrasian models. Simple applications for those two cases, as well as for linear and quadratic programming, are also provided.

1. Introduction

This paper provides a computer program written in *GAUSS* (version 2.0 or later) that can be used to solve linear and nonlinear complementarity problems. It can be used, in particular, to find equilibria for applied general equilibrium models, together with fix-price equilibria for dis-equilibrium models. Simple applications of these two cases are also provided in the paper.

The next section introduces the linear complementarity problem, a fundamental problem that arises frequently in several areas. The algorithm presented here is taken from Lemke (1965) and Ravindran (1972). This

section also presents the application of the algorithm to linear and quadratic programming. Section 3 presents its nonlinear counterpart: the so-called nonlinear complementarity problem, both in its classical and generalized versions, and an algorithm to solve it. Section 4 shows in turn, following Mathiesen (1985a, b), how to cast applied general equilibrium models in a nonlinear complementarity framework, and also solves a particular example. Section 5 does the same for the case of general fix-price models, and solves a simple model *à la* Barro-Grossman-Malinvaud. Finally, Appendix A presents the computer code of the program, while Appendices B and C present the results for the models in sections 4 and 5.

A final point before starting: The program was purportedly written to be used for problems in general form (Walrasian and non-Walrasian), and also to be as compact as possible. It was originally meant to be used in the classroom, as a way to show to graduate students in Economics the basics of the computation of Walrasian and non-Walrasian equilibria. For some particular applications, however, the reader will surely be able to enhance the performance of the program by modifying it in an *ad-hoc* fashion. For instance, in the case of general equilibrium models, Mathiesen (1985a, b) shows how to take advantage of the sparsity of a matrix that plays a key role in the algorithm. To conclude, the reader is encouraged to modify the program in the case of large models and/or when computer time is a constraint.

2. An Algorithm to Solve Linear Complementarity Problems

This section presents a description of the linear complementarity problem, together with Lemke's (1965) algorithm for its solution. It is worth pointing out, however, that Lemke's method is not the only procedure available (see, e.g., Murty, 1974, and the references in Murty, 1988). We have chosen it simply because of its widespread popularity and easiness of implementation. In the rest of the section we illustrate the use of the program by applying it to linear and quadratic programming. For more applications of the linear complementarity problem, see the delightful book by Murty (1988).

2.1. *The Linear Complementarity Problem*

Consider the following problem:

(LCP) Given an $n \times 1$ vector q and an $n \times n$ matrix $\mathbf{M}$, find two $n \times 1$ vectors $w \geq 0$ and $z \geq 0$ such that $w = \mathbf{M}z + q$ and $w'z = 0$.

Because of its linear structure and the fact that the entries of w and z have to be complementary in the sense that $w_i z_i = 0$ $(i = 1, \ldots, n)$, the problem given above is called in the literature the linear complementarity problem (LCP). Given that optimization problems as common as linear programming, quadratic programming, and two-person nonzero sum games can be cast as linear complementarity problems, Cottle and Dantzig (1968) refer to (LCP) as the "fundamental problem".

In order to solve (LCP), our program *RESUELVE* contains a procedure based on Lemke's algorithm of solution, Lemke (1965) and Cottle and Dantzig (1968). Briefly, and borrowing freely from the terminology used in the simplex algorithm, the computational steps involved in Lemke's method can be described as follows: Let e be the $n \times 1$ vector of ones, and consider the related problem

(RP) $\quad w - \mathbf{M}z - ez_0 = q,\ w \geq 0,\ z \geq 0,\ z_0 \geq 0,\ w'z = 0,$

where z_0 is a new variable (scalar) added to the problem. Obviously, any solution with $z_0 = 0$ to (RP) is also a solution to (LCP).

Set now $z = 0$ in (RP). If $q \geq 0$, the solution is clearly given by $w = q$; otherwise, let q_r be the smallest negative component of q and set $z_0 = -q_r$. The initial basis is then made by z_0 and all the entries of the corresponding $w \geq 0$ except for w_r (which is obviously zero). Denoting by $\mathbf{B}$ the current basis matrix, the rest of the procedure can be described, quoting Tomlin (1978), as follows:

(1) If w_r (respectively z_r) has become non-basic, z_r (resp. w_r) will enter the basis.

(2) If the candidate to become basic is w_r, let $a = \mathbf{B}^{-1}e$. And if it is z_r, let $a = -\mathbf{B}^{-1}m_r$ where m_r is the r-th column of $\mathbf{M}$.

(3) If $\beta_p / \alpha_p = \text{Min}\{\beta_i / \alpha_i, \text{ for } \alpha_i > 0 \text{ and } i = 1, \ldots, n\}$, where $\beta = \mathbf{B}^{-1}q$, then the p-th entry in the current basis becomes non-basic.

(4) Stop if z_0 has become non-basic or $z_0 < \delta$, for some small $\delta > 0$. Otherwise update $\mathbf{B}^{-1}$ and go to (1).

This procedure is simple enough, but care has to be taken in the updating of $\mathbf{B}^{-1}$, the choice of the pivot, and other issues reviewed by Tomlin (1978). We have chosen to write Lemke's procedure along the *FORTRAN* version proposed by Ravindran (1972) (see also Proll, 1974). Although his algorithm is not as robust as Tomlin's, its simplicity convinced us to use it as a blueprint for the *GAUSS* procedure given in Appendix A under the name *LEMKE*.

Another warning before turning to applications: Lemke's procedure is not guaranteed to process a solution for all matrices **M**. It always computes a solution, however, in the case of a copositive matrix (i.e., $x'\mathbf{M}x \geq 0$ for all $x \geq 0$), as well as in other common cases (see, e.g., Murty, 1988).

2.2. *Application to Linear Programming*

As the first example of a model that can be casted into a linear complementarity framework, consider the classical linear programming problem:

$$\text{Minimize } c'z$$

$$\text{s.t. } Ax \geq b, \quad x \geq 0,$$

where x, the primal vector, and c are of order $n \times 1$, A is of order $m \times n$, and b is of order $m \times 1$.

In order to solve this problem via the *LEMKE* procedure, one has to express it first as a linear complementarity problem. This can be accomplished as follows: If y denotes the dual vector, and u and v denote the primal and dual slackness vectors, then, after defining

$$w = \begin{pmatrix} v \\ u \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 0 & -A' \\ A & 0 \end{pmatrix}, \quad z = \begin{pmatrix} x \\ y \end{pmatrix}, \quad q = \begin{pmatrix} c \\ -b \end{pmatrix}$$

it follows that the linear programming problem can be written as a linear complementarity problem of the form (LCP). This is so because, by definition of the slack vectors, w has to be equal to $\mathbf{M}z + q$, and also because the complementary slackness conditions of linear programming imply that w and z are orthogonal.

The following lines in *GAUSS* solve a particular linear programming problem after calling the procedure *LEMKE*:

```
new;
let a = {3 6 -1 2,
        -4 2 1 5};
let b = {4,2};
let c = {6,20,3,20};
m = (zeros(4,4)~-a')|(a~zeros(2,2));
q = c|-b;
imax = 100;
{z,w} = lemke(m,q,imax);
```

where imax gives the maximum number of iterations allowed. The answer up to two decimals is: $z^* = (0, .62, 0, .15, 2.31, 3.08)$ and $w^* = (11.38, 0, 2.23, 0, 0, 0)$.

2.3. *Application to Quadratic Programming*

As a second application, consider the quadratic programming problem:

$$\text{Minimize } c'x + x'\mathbf{P}x$$

$$\text{s.t. } \mathbf{A}x \geq b, \; x \geq 0,$$

where x and c are of order $n \times 1$, $\mathbf{A}$ is of order $m \times n$, b is of order $m \times 1$, and $\mathbf{P}$ is an $n \times n$ symmetric positive semidefinite matrix.

In order to cast the problem into a linear complementarity framework, we just have to note that if x^* solves the quadratic problem, then, as shown for instance in Murty (1988), it also solves the following linear programming problem:

$$\text{Minimize } (c' + x^{*\prime}\mathbf{P})x$$

$$\text{s.t. } \mathbf{A}x \geq b, \; x \geq 0.$$

Thus, using now the trick employed in the last subsection, we end up with a linear complementarity problem after defining:

$$w = \begin{pmatrix} v \\ u \end{pmatrix}, \quad M = \begin{pmatrix} 0 & -A' \\ A & 0 \end{pmatrix}, \quad z = \begin{pmatrix} x \\ y \end{pmatrix}, \quad q = \begin{pmatrix} c \\ -b \end{pmatrix}.$$

The following lines in *GAUSS* state a particular, somewhat pathological, quadratic programming problem, and then call the procedure *LEMKE* to solve it:

```
new;
let a = {2 3 1 0,
        -2 -3 -1 0,
         1 4 0 1,
        -1 -4 0 -1};
let b = {6,-6,5,-5};
let c = {-1,-2,0,0};
p = (eye(2)~zeros(2,2))|zeros(2,4);
m = (p~-a')|(a~zeros(4,4));
q = c|-b;
imax = 100;
{z,w} = lemke(m,q,imax);
```

The answer, which took 6 iterations, is (up to two decimals): $z^* =$ (.76,1.06,1.29,0,0,0,0,.24) and $w^* =$ (0,0,0,.24,0,0,0,0).

3. An Algorithm to Solve Nonlinear Complementarity Problems

The main component of the program *RESUELVE* is a procedure with the same name that solves a generalization of the nonlinear complementarity problem. Before going through the most general case, however, it will be helpful to first consider the classical nonlinear complementarity problem. Namely,

(NCLP) Given a differentiable vector function $G: \mathbf{R}^n \to \mathbf{R}^n$, find a vector z, $z \geq 0$, such that $G(z)'z = 0$ and $G(z) \geq 0$.

As surveyed by Harker and Pang (1990), there are many applications that lead to a framework like (NLCP). In the case of Economics, it is to Mathiesen (1985a, b) credit to have realized that computable general equilibrium (CGE) models can be cast into this framework. In fact, Mathiesen's method has become a popular algorithm to solve CGE

models, for, as noted by Preckel (1985), it is faster than other procedures currently in use (of the kind described for instance in Scarf, 1985).

Once one has an algorithm to solve the linear complementarity problem, such as the one given last section, a procedure to solve (NLCP) can be easily implemented by transforming, using for instance Newton's method, the nonlinear complementarity problem to a sequence of linear complementarity problems.[1] More precisely, an algorithm to solve (NLCP) can be described as follows:

(1) Set $k = 0$ and specify an initial guess $z = z^0$.
(2) Set $k = k + 1$ and linearize $G(z)$ around z^{k-1}:
$L(G, z^{k-1}) \equiv q^k + \mathbf{M}^k z$, where $\mathbf{M}^k$ is the Jacobian matrix of G evaluated at z^{k-1}, and $q^k = G(z^{k-1}) - \mathbf{M}^k z^{k-1}$.
(3) Using Lemke's method, find z^k and w that solve
$w = \mathbf{M}^k z^k + q^k$, $w'z^k = 0$, $w \geq 0$, and $z^k \geq 0$.
(4) Stop if $| z^k - z^{k-1} | < \delta$, for some small $\delta > 0$. Otherwise go to (2).

The next section shows how computable general equilibrium models can be casted into this framework. For these models, Mathiesen (1985a,b, and 1987) and Rutherford (1989) provide refinements to the algorithm given above. It should be noted, however, that although it is quite easy to implement in *GAUSS* this procedure, we do *not* do so in our program. The reason is that, as first noted by Lensberg (1983), there is a more general version of the nonlinear complementarity problem that can be used not only to study Walrasian equilibrium models, but also non-Walrasian ones. This generalized nonlinear complementarity problem can be stated as:

(GNLCP) Given a differentiable vector function $F: \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$, find the $n \times 1$ vectors x and w, $x \geq 0$ and $w \geq 0$, such that $F(x, w) = 0$ and $x'w = 0$.

Following Lensberg (1983) and Lensberg and Rasmussen (1986), an algorithm to solve this problem can also be implemented by transforming it to a sequence of linear complementarity problems. But before introducing the algorithm, some notation is in order. Let

[1] For a different and promising approach to solve the nonlinear complementarity problem, see Harke and Xiao (1990).

$N \equiv \{1, 2, \ldots, 2n\}$, and let $v \in \mathbf{R}^n \times \mathbf{R}^n$ be defined as $v_i = x_i$ and $v_{i+n} = w_i$, $i = 1, \ldots, n$. For all $v \geq 0$ and such that $x'w = 0$, let $I(v)$ denote the set of n indices $i \in N$ for which $v_i > 0$ (if both x_i and w_i are zero then, say, let i be in $I(v)$). Thus, $I(v)$ contains the indices of the basic variables, while $N \setminus I(v)$ contains the indices of the non-basic ones. The main steps of the algorithm to solve (GNLCP) can then be described as follows:

(1*) Set $k = 0$ and specify an initial guess $v = v^0$ (such that the corresponding x^0 and w^0 are orthogonal).

(2*) Set $k = k + 1$ and linearize $F(v)$ around v^{k-1}: $L(F, v^{k-1}) \equiv c + \mathbf{M}v$, where $\mathbf{M}$ is the Jacobian matrix of F evaluated at v^{k-1}, and $c = F(v^{k-1}) - \mathbf{M}v^{k-1}$.

(3*) Obtain the partitions $\mathbf{M} = (\mathbf{M}_I, \mathbf{M}_{NI})$ and $v = (v_i, v_{NI})$ using the set of indices $I(v^{k-1})$.

(4*) Evaluate $L^k(F, v^{k-1}) \equiv \mathbf{M}_I^1 L(F, v^{k-1}) = c^k + v_I + \mathbf{M}^k v_{NI}$, where $c^k = \mathbf{M}_I^1 c$ and $\mathbf{M}^k = \mathbf{M}_I^1 \mathbf{M}_{NI}$.

(5*) Using Lemke's method, find v^k that solves: $v_I = -\mathbf{M}^k v_{NI} - c^k$, $v_I' v_{NI} = 0$ and $v \geq 0$.

(6*) Stop if $\sum_{i=1}^{n} | F^i(v^k) | < \delta$, for some small $\delta > 0$. Otherwise go to (2*).

The algorithm given above is implemented in *GAUSS* as the procedure *RESUELVE* in Appendix A. In section 5 it is shown, and exemplified, how fix-price models can be stated neatly in a context such as (GNLCP); but before that, the next section deals with the case of Walrasian models.

4. Solving Computable General Equilibrium Models

We now illustrate how to solve computable general equilibrium models by first stating the procedure in an abstract setting, and then solving a particular example. In this section we assume that the reader is already familiar with the relevant microeconomic concepts (otherwise, see, for instance, Shoven and Whalley, 1984).

Following Mathiesen (1985a, b), computable general equilibrium models can be cast as nonlinear complementarity problems by using an

activity analysis framework *à la* Scarf. Thus, consider an economy in which there are m activities and n goods, and let

$y = (y_1, \ldots, y_m)'$ be the vector of activity levels,
$p = (p_1, \ldots, p_n)'$ be the vector of good prices,
$b = (b_1, \ldots, b_n)'$ be the vector of initial endowments,
$d(p) = (d_1(p), \ldots, d_n(p))'$ be the vector of demand for goods, and
$\mathbf{A}(p) = [a_{ij}(p)]$ be the input-output matrix of order $m \times n$.

Note that the entries of the input-output matrix depend in general on prices (except for Leontieff technologies), and that positive (negative) entries in the matrix denote outputs (inputs).

In this framework, a competitive equilibrium is made of a vector of prices p^* and a vector of activity levels y^* such that:

i) $-\mathbf{A}(p^*)p^* \geq 0$, no activity earns positive profits;[2]
ii) $b + \mathbf{A}(p^*)'y^* - d(p^*) \geq 0$, no commodity is in excess demand;
iii) $p^* \geq 0$, $y^* \geq 0$, no prices or activity levels are negative;
iv) $[-\mathbf{A}(p^*)p^*]'y^* = 0$, no activity with negative profits is run;
v) $[b + \mathbf{A}(p^*)'y^* - d(p^*)]'p^* = 0$, the value of excess supply is zero.

If one defines the vector $z = (y_1, \ldots, y_m, p_1, \ldots, p_n)$, and the vector function $G(z)$ as $(G^y(z)', G^p(z)')'$, where

$$G^y(z) = -\mathbf{A}(p)p$$

$$G^p(z) = b + \mathbf{A}(p)'y - d(p),$$

it is then clear that the conditions for a competitive equilibrium that were stated above can be rephrased as: $z^* = (y^{*\prime}, p^{*\prime})'$ solves the nonlinear complementarity problem

$$G(z)'z = 0 \text{ with } z \geq 0 \text{ and } G(z) \geq 0.$$

[2] We are assuming technologies that exhibit constant returns to scale. As usual, if there were decreasing returns in some activity, then one could always add an artificial input whose factor payments would be equal to the profits of that activity.

Note that, since only relative prices matter in general equilibrium models, one has to choose a numeraire (and drop an equation) before attempting to solve the model.[3] This will be illustrated below.

As an alternative to the transformation given above, let us now recast the general equilibrium model into the generalized nonlinear complementarity framework (GNLCP) defined last section. Let w denote the vector $(w_1', w_2')'$, where w_1 and w_2 are nonnegative vectors of order $m \times 1$ and $n \times 1$, respectively. Let x denote, on the other hand, the vector $(y', p')'$. Finally, define the vector function $F(x, w)$ as $(F^1(x, w)', F^2(x, w)')'$, where

$$F^1(x, w) = w_1 + \mathbf{A}(p)p,$$

$$F^2(x, z) = w_2 - b - \mathbf{A}(p)'y + d(p).$$

It is clear then that any competitive equilibrium vector of activity levels and prices, x^*, must solve the generalized nonlinear complementarity problem

$$F(x, w) = 0, \text{ with } x \geq 0, w \geq 0, \text{ and } x'w = 0.$$

This way of recasting a general equilibrium is now exemplified by solving a very simple model presented in Shoven and Whalley (1984). In order to avoid lengthy expressions, we refer to their paper and to Appendix B below for the detailed equations. Here we simply point out how to rephrase their model in terms of the concepts discussed above.

On the production side, their model contains two final goods (1 and 2), produced using labor and capital (K and L), and using constant elasticity of substitution (CES) production functions. Denote by $y = (y_1, y_2)'$ the vector of activity levels, by $p = (p_1, p_2, p_L, p_K)'$ the vector of prices, and by $L_i(p, y_i)$ and $K_i(p, y_i)$ the conditional factor demands in sector i, given the activity level y_i. Consequently, the input-output matrix $\mathbf{A}(p)$ in this model can be expressed as

[3] The choice of the numeraire can also influence the speed of convergence (or lack of it). See Mathiesen (1985b, and 1987) and Rutherford (1989).

$$\mathbf{A}(p) = \begin{pmatrix} 1 & 0 & -L_1(p, 1) & -K_1(p, 1) \\ 0 & 1 & -L_2(p, 1) & -K_2(p, 1) \end{pmatrix}$$

On the other hand, Shoven and Whalley's model has two consumers (1 and 2) with CES utility functions depending only on the two final goods. The first (the "rich") consumer owns all the capital endowment, while the other (the "poor") owns all labor. Using the value of the parameters given in Table 1 of Shoven and Whalley (1984), the generalized nonlinear complementarity framework given above, and the price of labor as numeraire, Appendix B presents the computer code and the output obtained using *RESUELVE*. As can be seen there, the program is able to replicate the results in Shoven and Whalley's Table 2.

Granted, by casting the CGE model into a (GNLCP), rather than a (NLCP), the dimension of the Jacobian matrix is increased (and hence the computer time for numerical differentiation). But that is the price that one has to pay to be able to have an algorithm that also solves other models, such as the general fix-price models to be reviewed next.[4]

5. Solving Computable General Disequilibrium Models

In this last section we will now discuss fix-price models. The first thing to note is that a general disequilibrium model can be usually given a *dual* formulation in terms of the shadow prices associated with the quantity constraints in the primal (see Hahn, 1978, Laroque, 1981, and Lensberg, 1983). For instance, if there is rationing in a market due to a fixed price of the good, then there are implicitly two shadow prices on the binding constraint: the buyer's shadow price p_d, and the seller's shadow price p_s. But since at most only one side will be rationed, then it follows that p_d and p_s are complementary: $p_d p_s = 0$. As a consequence, if $\bar{p}$ denotes the fixed price prevailing in the market, then the virtual price that would make excess demand to be zero in the case of the buyer (or seller) would be given by $\bar{p} + p_d$ (or $\bar{p} - p_s$). Of course, this discussion can be trivially extended to the case of models that only exhibit downwards (or upwards) price stickiness.

[4] Needless to say, the reader that is only interested on solving CGE's will find it more convenient to simplify the first 25 lines or so of the program along the simpler (NLCP).

In order to exemplify what we have said, and also as an example of how a general disequilibrium model can be casted into a generalized nonlinear complementarity framework, we now consider a very simple fix-price model, in the spirit of Barro and Grossman (1971) and Malinvaud (1977), that is presented by Lensberg and Rasmussen (1986).

The first sector in the model involves a representative firm that uses labor to supply a single good (y_s). Labor is inelastically supplied up to a fixed amount of $\bar{L}$ and the production function is assumed to be given by:

$$y_s = L_d^a, \tag{5.1}$$

where L_d denotes the firm's demand for labor and a is less than one.

In this fix-price model there are, both, a rigid nominal price of labor, $\bar{w}$, and a fixed nominal price of output, $\bar{p}$, at each short-run equilibrium. Thus, in principle, the firm could be constrained in both the labor and the output markets. Nevertheless, if w_d and p_s denote the firm's shadow prices of the constraints on labor and output, then, as noted earlier, the virtual prices $\bar{p} - p_s$ and $\bar{w} + w_d$ can be used to find the demand for labor and the supply of the good that maximize profits regardless of whether or not those constraints are binding. It is easy to show that they are:

$$L_d = [a(\bar{p} - p_s) / (\bar{w} + w_d)]^{1/(1-a)}, \tag{5.2}$$

$$y_s = [a(\bar{p} - p_s) / (\bar{w} + w_d)]^{a/(1-a)}. \tag{5.3}$$

The other sector of the model is made of a representative household that supplies labor inelastically, and demands goods (y_d) together with money (M_d). It should be noted that we will assume that the money market is always in equilibrium, so that we can drop the subindex of M. The household's utility function is assumed to be given by:

$$U(y_d, M/\bar{p}) = r \ln (y_d) + (1 - r) \ln (M/\bar{p}), \tag{5.4}$$

where r is the budget share for commodities. Given initial money holdings M_0, the household's budget constraint is given by

$$\bar{p} y_d + M \leq \bar{p} y_s + M_0. \tag{5.5}$$

As in the case of the firm, the household could end up being rationed. Let p_d denote the household's shadow price of the (eventual) rationing constraint on the goods market, so that the virtual price of the good is given by $\bar{p}+p_d$. Using this price, let I be the household's virtual income. Then the optimal demands for goods and money are given by:

$$y_d = rI/(\bar{p}+p_d), \tag{5.6}$$

$$M = (1-r)I \tag{5.7}$$

We now proceed to find the non-Walrasian equilibrium quantities, together with the shadow prices on the rationing constraints, by casting the computational problem as a generalized nonlinear complementarity problem. This can be acomplished as follows: First, let *ES* denote the excess supply of goods

$$ES \equiv y_s - y_d = [a(\bar{p}-p_s)/(\bar{w}+w_d)]^{a/(1-a)} - rI/(\bar{p}+p_d), \tag{5.8}$$

which must equal zero at the equilibrium virtual prices.

Secondly, if u denotes the number of units of time that the household is unemployed, then

$$ED \equiv L_d + u - \bar{L} = [a(\bar{p}-p_s)/(\bar{w}+w_d)]^{1/(1-a)} + u - \bar{L} \tag{5.9}$$

has to be by definition equal to zero; furthermore, u and w_d are complementary.[5]

Finally, let *EI* denote the household's unused income:

$$EI \equiv \bar{p}y_s + M_0 - \bar{p}y_d - M + s = \bar{p}(y_s - y_d) + M_0 - (1-r)I + s \tag{5.10}$$

which is zero at the optimum. We have added the slack variable s, which is also equal to zero at the solution, in order to have a complementary variable for I.

Given values for the parameters $\bar{p}, \bar{w}, a, r, \bar{L}$ and M_0, a fix-price equilibrium (p_s, w_d, I, p_d, u, s) is obtained if (1) the three functions

[5] We have to resort to this trick since the household supplies labor inelastically, and hece there is no shadow price on the eventual rationing constraint.

(5.8)-(5.10) evaluated at this point are equal to zero, and (2) the corresponding complementarity conditions are fulfilled:

$$p_s p_d = 0, \tag{5.11}$$

$$w_d u = 0, \tag{5.12}$$

$$sI = 0, \tag{5.13}$$

$$p_s, w_d, I, p_d, u, s \geq 0. \tag{5.14}$$

The set of equations (5.8)-(5.14), constitute a generalized nonlinear complementarity problem. This is illustrated in Appendix C where, given the parameters $\bar{p} = 2$, $\bar{w} = 1$, $a = 0.45$, $r = 0.8$, $\bar{L} = 1$ and $M_0 = 0.5$, the solution is found to be (up to two decimal points): $p_s = 0$, $w_d = 0$, $I = 2.50$, $p_d = 0.18$, $u = 0.17$, and $s = 0$. This coincides with the solution reported by Lensberg and Rasmussen (1986).

Appendix A: Computer Code for the Program *RESUELVE*

The main component of the program written in *GAUSS* (version 2.0 or later) is made of the procedure *RESUELVE*. This procedure invokes in turn the procedure *LEMKE*, given also below. Note that the latter can be used independently in the case of linear complementarity problems. The procedure *RESUELVE*, as exemplified in the other appendices, requires an initial guess for the vector "v", an initial basis ("basis"), a maximum number of iterations to be allowed ("imax"), and a procedure that defines the vector function ("f"). The procedure *LEMKE* only requires the matrix "m", the vector "q", and the corresponding "imax". Including output instructions, the entire program is made of 126 instructions (a diskette containing the computer code is freely available upon request from the author).

```
proc resuelve(v,basis,imax);
    local d,g,gb,gbinv,gnb,i,k,m,n,nbasis,q,vb,vnb;
    n = rows(basis);
    i = 0;
    do while i < imax;
```

```
            i = i+1;
            print; print "ITERATION";; format /rd 3,0; i;
            format /rd 8,2; print "v' =";; v'; print "F(v)' =";; f(v)';

/* Linearize the nonlinear problem, and call lemke */
      g = gradp(&f,v);
      gb = submat(g,0,basis);
      d = basis . > n*ones(n,1);
      nbasis = substute(basis+n*ones(n,1),d,basis-n*ones(n,1));
      gnb = submat(g,0,nbasis);
      gbinv = inv(gb);
      m = -gbinv*gnb;
      q = -gbinv*(f(v)-g*v);
      print "PROC LEMKE PROCEDURE IS CALLED... ";;
      {vnb,vb} = lemke(m,q,imax);
      k = 0;
      do while k < n;
            k = k+1;
            if vnb[k] > 0;
                  v[nbasis[k]] = vnb[k];
                  basis[k] = nbasis[k];
            else;
                  v[basis[k]] = vb[k];
                  endif;
            endo;
            if sumc(abs(f(v))) <= 1E-10;
                  print; print "SOLUTION FOUND:";
                  format /rd 8,2; print "v' =";; v'; print "F(v)' =";; f(v)';
                  retp(v);
            endif;
      endo;
      print; print "NO SOLUTION AFTER";; format /rd 3,0; imax;;
      print "ITERATIONS";
endp;

proc (2) = lemke(m,q,imax);
      local alpha,b,basic,d,in,j,k,n,out,r,r0,w,z,z0;
      n = rows(m);

/* Introduce the new variable z0 and get the basis */
      z0 = -minc(q);
      r = minindc(q);
      z = zeros(n,1);
      if z0 < 0;
            w = q;
            print "TRIVIAL SOLUTION TO LCP: w = q, z = 0";
            retp(z,w);
```

```
        endif;
        q = q+z0;
        q[r] = z0;
        b = eye(n);
        b[.,r] = - ones(n,1);
        basic = ones(n,1)|seqa(1,1,n);
        basic[r] = 0;
        basic[r+n] = 0;
        r0 = r;
        w = q;
        w[r] = 0;
        out = 1|r;
        j = i;

/* Find the new basis column to enter or detect solution */
d1: if out[1] = = 1;
            in = 2|out[2];
            alpha = -b*m[.,in[2]];
        elseif out[1] == 2;
            in = 1|out[2];
            alpha = b[.,in[2]];
        else;
d2:         print "SOLUTION TO LCP (AFTER";; format /rd 3,0; j;;
            print "ITERATIONS):";
            d = q . <= 1E-10*ones(n,1);
            q = substute(q,d,0);
            w = zeros(n,1);
            z = zeros(n,1);
            k = 0;
            do while k < n;
                k = k+1;
                if basic[k] == 1;
                    w[basic[k+n]] = q[k];
                elseif basic[k] == 2;
                    z[basic[k+n]] = q[k];
                endif;
            endo;
            format /rd 8,2; print "z' =";; z'; print "w' =";; w';
            retp(z,w);
        endif;

        /* Find the pivot row for next iteration */
        if q[r0] <= 1E-10;
            goto d2;
        endif;
        if alpha <= zeros(n,1);
            print "NO SOLUTION TO LCP";
```

```
        end;
    else;
        r = maxindc(alpha);
    endif;
    k = 0;
    do while k < n;
        k = k+1;
        if alpha[k] <= 0;
            continue;
        elseif q[k]/alpha[k] < q[r]/alpha[r];
            r = k;
        endif;
    endo;

/* Make pivot operation and update the basis and q */
    b[r,.] = b[r,.]/alpha[r];
    q[r] = q[r]/alpha[r];
    k = 0;
    do while k < n;
        k = k+1;
        if k /= r;
            q[k] = q[k]-q[r]*alpha[k];
            b[k,.] = b[k,.]-b[r,.]*alpha[k];
        endif;
    endo;
    out[1] = basic[r];
    out[2] = basic[r+n];
    basic[r] = in[1];
    basic[r+n] = in[2];
    if j < imax;
        j = j+i;
        goto d1;
    else;
        print "NO SOLUTION TO LCP AFTER";; format /rd 3,0; imax;;
        print "ITERATIONS";
        end;
    endif;
endp;
```

Appendix B: Program and Output for the Model in Section 4

In order to solve Shoven and Whalley's model (1984), we choose as numeraire labor and drop the equation for its excess supply (i.e., $b_L - L_1 y_1 - L_2 y_2$). Next, let $v' = (y_1, y_2, p_1, p_2, p_k, w_1, w_2, w_3, w_4, w_5)$,

where all the prices are now relative prices. The computer code that defines the corresponding function $F(v)$ is given in the procedure "proc $f(v)$" below. Finally, before calling *RESUELVE* an initial guess for the basis is required. In the case of *CGE*'s it is natural to guess that the basis is made of all variables except the w's, since these artificial variables are greater than zero only if there is excess supply or positive profits. Thus the initial basis is $(y_1, y_2, p_1, p_2, p_k)$ with (arbitrary) values (10, 10, 1, 1, 1).

```
new;
      let v = {10,10,1,1,1,0,0,0,0,0};
      let basis = {1,2,3,4,5};
      imax = 100;
      call resuelve(v,basis,imax);
end;

proc f(v);
      local fv,k1,k2,l1,l2,x11,x12,x21,x22;
      fv = zeros(5,1);
      k1 = 1/(1.5*(.9*v[5]+.4)^2);
      k2 = .15+(.0525/v[5])^.5;
      l1 = .24*v[5]^2/(.36*v[5]+.16)^2;
      l2 = .35+(.0525*v[5])^.5;
      x11 = 25*v[5]/(v[3]+(v[3]^1.5)/(v[4]^.5));
      x12 = 18/(.3*v[3]+.7*(v[3]^.75)*(v[4]^.25));
      x21 = 25*v[5]/(v[4]+(v[4]^1.5)/(v[3]^.5));
      x22 = 42/(.7*v[4]+.3*(v[4]^.75)*(v[3]^.25));
      fv[1] = v[6]+v[3]-l1-v[5]*k1;
      fv[2] = v[7]+v[4]-l2-v[5]*k2;
      fv[3] = v[8]-v[1]+x11+x12;
      fv[4] = v[9]-v[2]+x21+x22;
      fv[5] = v[10]-25+v[1]*k1+v[2]*k2;
      retp(fv);
endp;
```

The rest of this appendix reproduces the output obtained after running the program (using a format with two decimal points):

```
ITERATION 1
v'      =      10.00 10.00      1.00     1.00     1.00     0.00     0.00     0.00
     0.00           0.00
F(v)'   =      -0.28     0.04   20.50    44.50   -17.26
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
```

```
ITERATION 2
v'      =     22.16    52.34    1.50    1.16    1.54    0.00    0.00    0.00
      0.00          0.00
F(v)'   =      0.06     0.01    2.49    0.66   -2.87
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
ITERATION 3
v'      =     24.67    54.43    1.39    1.08    1.33    0.00    0.00    0.00
      0.00          0.00
F(v)'   =      0.01     0.00    0.00    0.11    0.42
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
ITERATION 4
v'      =     24.93    54.38    1.40    1.09    1.37    0.00    0.00    0.00
      0.00          0.00
F(v)'   =      0.00     0.00   -0.00    0.00    0.01
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
ITERATION 5
V'      =     24.94    54.38    1.40    1.09    1.37    0.00    0.00    0.00
      0.00          0.00
F(v)'   =      0.00     0.00    0.00    0.00    0.00
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z= 0
SOLUTION FOUND:
v'      =     24.94    54.38    1.40    1.09    1.37    0.00    0.00    0.00
      0.00          0.00
F(v)'   =      0.00     0.00    0.00    0.00    0.00
```

Appendix C: Program and Output for the Model in Section 5

In order to solve the system of equations (5.8)-(5.14), one has to recast it into a generalized nonlinear complementarity framework. Let $v = (p_s, w_d, I, p_d, u, s)$, in such a way that the complementarity conditions are given by $v_i v_{i+3} = 0$ $[i \leq 3]$. Let also $F(v) = (ES(v), ED(v), EI(v))$. The computer code that defines this function is given in the procedure "proc f(v)" below. Finally, before calling *RESUELVE* an initial guess for the basis is required. In our example, the guess was (p_s, u, I) with values (0,0,2).

```
new;
      let v = {0,0,2,0,0,0};
      let basis = {1,3,5};
      imax = 100;
      call resuelve(v,basis,imax);
end;
```

```
proc f(v);
      local a,fv,ld,ls,m0,pf,r,ys,yd,wf;
      fv = zeros(3,1);
      pf = 2;
      wf = 1;
      a = 0.45;
      r = 0.8;
      ls = 1;
      m0 = 0.5;
      ld = (a*(pf-v[1])/(wf+v[2]))^(1/(1-a));
      ys = ld^a;
      yd = r*v[3]/(pf+v[4]);
      fv[1] = ys-yd;
      fv[2] = ld+v[5]-ls;
      fv[3] = pf*ys+m0-pf*yd-(1-r)*v[3]+v[6];
      retp(fv);
endp;
```

The rest of this appendix reproduces the output obtained after running the program (using a format with two decimal points):

```
ITERATION 1
v'      =     0.00     0.00     2.00     0.00     0.00     0.00
F(v)'   =     0.12    -0.17     0.33
PROC LEMKE PROCEDURE IS CALLED... SOLUTION TO LCP
(AFTER 2 ITERATIONS):
z'   =   0.21     0.00     0.00
w'   =   0.00     2.50     0.17
ITERATION 2
v'      =     0.00     0.00     2.50     0.21     0.17     0.00
F(v)'   =     0.01    -0.00     0.02
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
ITERATION 3
v'      =     0.00     0.00     2.50     0.18     0.17     0.00
F(v)'   =    -0.00     0.00    -0.00
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
ITERATION 4
v'      =     0.00     0.00     2.50     0.18     0.17     0.00
F(v)'   =    -0.00     0.00    -0.00
PROC LEMKE PROCEDURE IS CALLED... TRIVIAL SOLUTION TO LCP: w = q, z = 0
SOLUTION FOUND:
v'      =     0.00     0.00     2.50     0.18     0.17     0.00
F(v)'   =     0.00     0.00     0.00
```

References

Barro, R. J., and H. I. Grossman (1971). "A General Disequilibrium Model of Income and Employment", *American Economic Review*, 1971, 61, 82-93.

Cottle, R. W., and G. B. Dantzig (1968). "Complementary Pivot Theory of Mathematical Programming", *Linear Algebra and its Applications*, 1, 103-125.

Hahn, F. H. (1978). "On Non-Walrasian Equilibria", *Review of Economic Studies*, 45, 1-18.

Harker, P. T. (1993). *Lectures on Computation of Equilibria with Equation-Based Methods*, Louvain, Belgium: CORE.

———, and J. Pang (1990). "Finite-Dimensional Variational Inequality and Nonlinear Complementarity Problems: A Survey of Theory, Algorithms and Applications", *Mathematical Programming*, 48, 161-220.

———, and B. Xiao (1990). "Newton's Method for the Nonlinear Complementarity Problem: A B-Differentiable Equation Approach", *Mathematical Programming*, 48, 339-357.

Laroque, G. (1981). "On the Local Uniqueness of Fixed Price Equilibria", *Review of Economic Studies*, 48, 113-129.

Lemke, C. E. (1965). "Bimatrix Equilibrium Points and Mathematical Programming", *Management Science*, 11, 681-689.

Lensberg, T. (1983). "A Dual Route to Computable Fix-Price Equilibria", Discussion Paper 0583, Norwegian School of Economics and Business Administration, Bergen, Norway.

———, and H. Rasmussen (1986). "COMPAK Version 2.0 User Manual", Working Paper MU 09, Norwegian School of Economics and Business Administration, Bergen, Norway.

Malinvaud, E. (1977). *The Theory of Unemployment Reconsidered*, Oxford: Basil Blackwell.

Mathiesen, L. (1985a). "Computation of Economic Equilibria by a Sequence of Linear Complementarity Problems", *Mathematical Programming Study*, 23, 144-162.

——— (1985b). "Computational Experience in Solving Equilibrium Models by a Sequence of Linear Complementarity Problems", *Operations Research*, 33, 1225-1250.

——— (1987). "An Algorithm Based on a Sequence of Linear Complementarity Problems Applied to a Walrasian Equilibrium Model: An Example", *Mathematical Programming*, 37, 1-18.

Murty, K. G. (1974). "Note on a Bard-type Scheme for Solving the Complementarity Problem", *Opsearch*, 11, 123-130.

——— (1988). *Linear Complementarity, Linear and Nonlinear Programming*, Berlin: Helderlmann Verlag.

Preckel, P. V. (1985). "Alternative Algorithms for Computing Economic Equilibria", *Mathematical Programming Study*, 23, 163-172.

Proll, L. G. (1974). "Remark on Algorithm 431", *Communications of the ACM*, 17, 590.

Ravindran, A. (1970). "A Computer Routine for Quadratic and Linear Programming Problems", *Communications of the ACM*, 15, 818-820.

Rutherford, T. F. (1989). "General Equilibrium Modelling with MPS/GE", manuscript, University of Western Ontario, London, Canada.

Scarf, H. (1985). "The Computation of Equilibrium Prices", in K. Arrow and M. Intriligator (eds.), *Handbook of Mathematical Economics*, vol. 2, New York: North Holland.

Shoven, J. B., and J. Whalley (1984). "Applied General Equilibrium Models of Taxation and International Trade: An Introduction and Survey", *Journal of Economic Literature*, 22, 1007-1051.

Tomlin, J. A. (1978). "Robust Implementation of Lemke's Method for the Linear Complementarity Problem", *Mathematical Programming Study*, 7, 55-60.