A Pathsearch Damped Newton Method for Computing General Equilibria

Steven P. Dirkse [*] Michael C. Ferris [*]

April, 1994

Abstract

Computable general equilibrium models and other types of variational inequalities play a key role in computational economics. This paper describes the design and implementation of a pathsearch-damped Newton method for solving such problems. Our algorithm improves on the typical Newton method (which generates and solves a sequence of LCP's) in both speed and robustness. The underlying complementarity problem is reformulated as a normal map so that standard algorithmic enchancements of Newton's method for solving nonlinear equations can be easily applied. The solver is implemented as a GAMS subsystem, using an interface library developed for this purpose. Computational results obtained from a number of test problems arising in economics are given.

1 Introduction

The variational inequality (VI) is of fundamental importance in computational economics and in many other disciplines as well. In economics, examples include general, Walrasian, Nash, and spatial price equilibrium models, as well as the optimality conditions for nonlinear programming. VI's are common in mathematical programming, game theory, and transportation and regional science as well.

Recently, a complementarity format has been incorporated into the GAMS language and been made available by GAMS Development Corporation. This format (called GAMS/MCP) makes use of the CPLIB library (Dirkse, Ferris, Preckel & Rutherford 1993) to formulate complementarity problems and pass them on to a complementarity solver. The computed solution is then returned to CPLIB and reported by GAMS. Rutherford (1994*b*) demonstrates the use of the GAMS/MCP system for equilibrium analysis and game theory models. In addition, Rutherford (1994*a*) has embedded MPSGE, a modeling language designed specifically

[*]Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706, email: *dirkse@cs.wisc.edu*, *ferris@cs.wisc.edu*. This material is based on research supported by the National Science Foundation Grant CCR–9157632 and the Air Force Office of Scientific Research Grant F49620–94–1–0036.

for solving Arrow-Debreu economic equilibrium models, in GAMS/MCP. An equilibrium model can be described at a higher level using MPSGE than is possible using GAMS/MCP alone, greatly reducing the work of model formulation. These two systems allow the equilibrium modeler to take advantage of the power and flexibility of the GAMS modeling language, while our solver, implemented as a GAMS subsystem, is available for solving such models.

A number of methods, surveyed by Harker & Pang (1990), have been proposed and used for computing solutions to variational and complementarity problems. Among the most powerful of these is the adaptation of Newton's method for variational inequalities due originally to Josephy (1979) (see also (Robinson 1993)). As in the smooth case, local quadratic convergence in a region containing a solution point can be shown under suitable conditions. Unfortunately, the method may fail to converge outside of this region, which may be quite small. Thus, the success of Newton's method, and Josephy's variant for VI, is very dependent on the starting point chosen. This lack of robustness in Newton's method for nonlinear equations has been reduced substantially through the use of linesearch techniques which globalize the domain of convergence (Ortega & Rheinboldt 1970).

The PATH solver is an extension of Josephy's variant of Newton's method for complementarity problems which includes a pathsearch component. This pathsearch makes use of a piecewise-linear path from the current iterate to the corresponding "Newton point", and generalizes the linesearch used in Newton's method to enlarge the region of convergence. In addition, our path construction technique, which is similar to the pivotal method proposed by Lemke & Howson (1964), significantly reduces the number of pivot steps necessary to identify the Newton point. The pathsearch damping depends on reformulating the problem as one of solving a square system of nonsmooth equations.

The purpose of this paper is to describe the design and implementation of a robust, general purpose, and efficient Newton method for solving large complementarity problems. Our implementation makes use of sparse matrix algebra to increase the dimension of the problems we can effectively solve. It also employs CPLIB to gain access to a large number of equilibrium and complementarity models which have been formulated in GAMS/MCP format (Dirkse & Ferris 1994*a*, Rutherford 1994*a*). We believe that the GAMS/MCP system, coupled with the PATH solver or with MILES (Anstreicher, Lee & Rutherford 1992, Rutherford 1994*b*), provide economists with more modeling and solving capability than has previously been available. While it is possible to derive convergence results for the PATH solver (Dirkse & Ferris 1994*b*), we will focus instead on the algorithm employed and the design and implementation issues involved.

In order to be solved by the PATH solver, a problem must be expressed as a Mixed Complementarity Problem, or MCP. This formulation, also known as the rectangular or box-constrained VI, represents a slight generalization of the standard non-negatively constrained nonlinear complementarity problem, and is described in Section 2.

Some definitions are in order. The set $\mathbb{R}^n_+$ denotes the positive orthant, or the set of points $x \in \mathbb{R}^n$ such that $x \geq 0$. Assuming the set $C \subset \mathbb{R}^n$ is closed and convex, we denote the projection operator onto the set C as $\pi_C(\cdot)$; $\pi_C(x)$ is the unique point in C which minimizes the Euclidean norm $\|c - x\|_2$ for $c \in C$. The projection of a vector x onto $\mathbb{R}^n_+$ is denoted more simply by x_+.

2 The Mixed Complementarity Problem

The Nonlinear Complementarity Problem (NCP) typically considered in the literature (Harker & Pang 1990) is that of finding z such that

$$F(z) \geq 0, \quad z \geq 0, \quad \langle F(z), z \rangle = 0, \tag{NCP}$$

where $F : \mathbb{R}^n_+ \to \mathbb{R}^n$ is a (usually) continuously differentiable function of the variable z. If we define a rectangular set or box $B := \{z \mid \ell \leq z \leq u\}$, where $\ell_i \in [-\infty, \infty)$, $u_i \in (-\infty, \infty]$, we see that the constraint $z \geq 0$ in the definition of NCP is a special instance of $z \in B$, where $\ell = 0$ and $u = \infty$. The Mixed Complementarity Problem (MCP) is the generalization of NCP to the case where the variable z is subject to box constraints.

Definition 1 (MCP) *Given a function $F : \mathbb{R}^n \to \mathbb{R}^n$ and bounds ℓ and u,*

$$\text{find} \quad z \in \mathbb{R}^n, \;\; w, v \in \mathbb{R}^n_+$$

$$\text{s. t.} \quad \begin{aligned} F(z) &= w - v && \text{(1a)} \\ \ell \leq &\, z \leq u && \text{(1b)} \\ (z - \ell)^\top w &= 0 && \text{(1c)} \\ (u - z)^\top v &= 0 && \text{(1d)} \end{aligned}$$

In this formulation, w and v are the positive and negative parts of $F(z)$, which must be complementary to the difference between z and its lower and upper bounds ℓ and u, respectively. Note that the choice of z completely determines w and v, so that we can speak of z solving MCP where convenient.

Many problems commonly considered in the literature are equivalent or can be reduced to MCP, including nonlinear equations ($B := \mathbb{R}^n$) and nonlinear complementarity problems. MCP reduces to NCP when the box B defined by ℓ and u is the positive orthant (i.e. $\ell := 0$ and $u := \infty$). These bounds imply that $z \geq 0$, while (1d) implies that $v \equiv 0$, so that $F(z) = w \geq 0$, while $\langle F(z), z \rangle = 0$ follows from (1c).

The variational inequality, or VI, has much in common with the MCP. Given a convex set $C \subset \mathbb{R}^n$ and a function $F : C \to \mathbb{R}^n$, VI(F, C) is defined as follows: find $z \in C$ such that

$$\langle F(z), (c - z) \rangle \geq 0, \quad \forall\, c \in C. \tag{VI}$$

If the feasible set C is rectangular, then VI(F, C) and the MCP defined by F and C are completely equivalent, as their solution sets are identical. A proof of this is elementary. When C is polyhedral rather than rectangular, VI(F, C) can be reduced to an MCP by explicitly including the dual variables to the constraints defining C. Thus, given a box B and a set $X := \{z \mid Az \leq b\}$, where $A \in \mathbb{R}^{m \times n}$, it can be shown that VI$(F, B \bigcap X)$ is equivalent to VI$(H, B \times \mathbb{R}^m_+)$, where

$$H(z, u) = \begin{bmatrix} F(z) + A^\top u \\ -Az + b \end{bmatrix}.$$

When equality constraints are used to define X, the associated dual variables u are free. Two advantages to using the MCP formulation as opposed to the NCP are the explicit treatment of simple bounds on the variables z and the availability of free variables, which enable the explicit representation of equality constraints. This is more efficient than introducing extra variables and equations to deal with general bounds and equality constraints.

In order to perform computational tests of our algorithm for solving MCP, we have formed MCPLIB, a library of complementarity models expressed in GAMS/MCP format (Dirkse & Ferris 1994*a*). These models come from a wide variety of disciplines, as indicated in Table 1. In addition to providing a convenient environment in which to test our solver, MCPLIB serves as an example of what is possible using the complementarity format present in GAMS, and provides a means by which other algorithm developers can test their own solvers and compare them with those already available.

Table 1: MCPLIB models

Model origin	GAMS file	Size
Nonlinear equations		
Distillation column modeling	hydroc*	39, 99
" "	methan08	39
Nonlinear programming		
NLP test problem from Colville (1968)	colvncp, colvdual	15, 20
Obstacle problems	obstacle, bratu	N
Nonlinear complementarity		
	josephy, kojshin	4
Elastohydrodynamic lubrication	ehl_kost	N
Variational inequalities		
Nash equilibrium	nash, choi	10, 14
Spatial price equilibrium	sppe, tobin	27, 42
Walrasian equilibrium	mathi*	4
" "	scarfa*, scarfb*	14, 40
Traffic assignment	bertseka, gafni	15, 5
Invariant capital stock	hanskoop	14
Project Independence energy system (PIES)	pies	42
Von Thünen land use	vonthun	186
Extended linear-quadratic programming		
Optimal control	opt_cont	N

In addition, Rutherford (1994*a*) has embedded MPSGE, a modeling language designed specifically for solving Arrow-Debreu economic equilibrium models, in GAMS/MCP. There is also a library of general equilibrium models which accompanies this. With MPSGE, an equilibrium model can be described at a higher level than that possible using GAMS/MCP alone. Although not as flexible as GAMS/MCP, MPSGE allows a much more concise definition of those models for which it is appropriate, thus avoiding the tedium and error associated with a more explicit definition.

As an example of how GAMS/MCP is used, we include the GAMS model for a simple Walrasian equilibrium problem given by Mathiesen (1987). The equilibrium conditions for this model are

$$
\begin{aligned}
b - d(p) + Ay \geq 0, &\qquad p \geq 0, \qquad \perp \qquad (2)\\
-A^{\top} p \geq 0, &\qquad y \geq 0, \qquad \perp, \qquad (3)
\end{aligned}
$$

where the demand function $d(\cdot)$ is defined by

$$
d_i(p) \ := \frac{a_i \ \sum_k b_k p_k}{p_i}
$$

and the data a, b and A are given. The GAMS model for this problem, without the obvious parameter definitions, is given in Figure 1. The key statement in the model in Figure 1 is

```
sets COM / 1 * 3 /,        /* commodities */
S / 1 /;                   /* production sectors/activities */
alias (COM,K);
parameters a(COM),         /* demand shares */
b(COM),                    /* endowments */
A(COM,S);                  /* activity analysis */
positive variables
p(COM),                    /* commodity prices */
y(S);                      /* activity levels */
equations
supply(COM),               /* excess */
profit(S);

supply(COM) ..    b(COM) -a(COM) * sum(K, b(K)*p(K)) / p(COM)
                  + sum(S, A(COM,S)*y(S)) =g= 0;
profit(S) ..      -sum(COM, p(COM)*A(COM,S)) =g= 0;
model mathiesen   / supply.p, profit.y /;
solve mathiesen using mcp;
```

Figure 1: A Walrasian model in GAMS/MCP format

the `model` statement. The dot notation is used there to indicate that a complementarity relationship must hold between the associated function–variable pair. The complementarity relationships determined by all function–variable pairs defined in the `model` statement, when coupled with the bounds defined for the variables, are equivalent to an MCP given by a function F and a box B. It is this F and B which are passed to a solver.

Our solver makes use of CPLIB, a subroutine library that acts as an interface to GAMS. Its primary function is to interpret the complementarity relationships defined by the model statement, verify that these relationships are valid, and provide the user with function, gradient, and bound information for the corresponding F and B. There are also routines for getting initial values of the problem variables and for reporting solution results, as well as a number of routines which enable the user to read and write model data and parameter values. In addition, CPLIB provides the Fortran programmer with the option of dynamically allocating one block of memory, to be used by the solution algorithm. This avoids the need to build any limits on problem size into a solver at compile time.

3 The PATH Solver

The PATH solver is an implementation of a stabilized Newton method for solving MCP which generalizes similar techniques used for solving nonlinear equations. The algorithm employed makes use of the path construction and searching techniques first explored by Ralph (1994) and later developed by Dirkse & Ferris (1994*b*). Before describing this algorithm, we review linesearch damped Newton's method for nonlinear equations. In the sequel, we shall assume that the function F is differentiable on the box B.

In the classical damped Newton method for solving

$$F(x) = 0 \tag{4}$$

(see for example Ortega & Rheinboldt (1970)), the smooth function F is approximated at a point x^k by a linearization A_k defined by

$$A_k(x) := F(x^k) + F'(x^k)(x - x^k). \tag{5}$$

The Newton point x_N^k is a zero of this approximation, i.e. $A_k(x_N^k) = 0$. If the Jacobian matrix $F'(x^k)$ is nonsingular, this zero is unique, and is conceptually easy to find. Upon solving the matrix equation $F'(x^k)d^k = -F(x^k)$, the Newton point is given by $x_N^k = x^k + d^k$, where d^k is the Newton direction. The next iterate in the Newton process is determined by a linesearch along this direction. The new point

$$x^{k+1} := x^k + \lambda d^k,$$

is chosen to satisfy some descent criteria in $\|F\|$, with the ultimate goal of finding a point x^* such that $\|F(x^*)\| = 0$. Such a point must solve $F(x^*) = 0$ as well.

In a damped Newton method for (4), the fact that a solution x^* is also a minimizer of $\|F\|$ is crucial. However, a solution z^* of MCP will not generally minimize $\|F\|$. In order to apply a damped Newton method, we must rewrite MCP as a zero-finding problem. To do so, note that since MCP and VI(F, B) are equivalent, z solves MCP if and only if

$$\langle -F(z), b - z\rangle \leq 0, \qquad \forall\, b \in B. \tag{6}$$

If we define $x := z - F(z)$, then the inequality

$$\langle x - z, b - z\rangle \leq 0, \qquad \forall\, b \in B \tag{7}$$

follows from (6). But (7) is the projection inequality (Hiriart-Urruty & Lamarechal 1993); assuming $z \in B$, (7) holds if and only if $z := \pi_B(x)$, the Euclidean projection of x onto B. Hence, the following equation

$$-F(\pi_B(x)) = x - \pi_B(x), \tag{8}$$

is satisfied. Conversely, if (8) holds, then since the projection inequality (7) holds for $z = \pi_B(x)$, it follows that (6) holds as well. Thus, $\pi_B(x)$ solves MCP, so that solving equation (8) is equivalent to solving MCP. This is made precise in the following definition and theorem.

Definition 2 (Normal Map) *Given a convex set $B \subset \mathbb{R}^n$ and a function $F : B \to \mathbb{R}^n$, the normal map $F_B(\cdot)$ induced on F by B is defined as*

$$F_B(x) := F(\pi_B(x)) + (x - \pi_B(x)).$$

The corresponding normal equation is then defined as

$$0 = F_B(x) = F(\pi_B(x)) + (x - \pi_B(x)). \tag{NE}$$

Thus we have shown the following result.

Theorem 3 *Given a rectangular set $B := \{z \mid \ell \le z \le u\}$ and function $F : B \to \mathbb{R}^n$, the vector $x \in \mathbb{R}^n$ solves NE $\Rightarrow z := \pi_B(x)$ solves MCP, while z solves MCP $\Rightarrow x := z - F(z)$ solves NE.*

Since the projection mapping π_B is continuous (Hiriart-Urruty & Lamarechal 1993), a necessary and sufficient condition for the continuity of F_B is the continuity of F on B. However, since π_B is nondifferentiable, F_B also fails to be differentiable. In order to better understand the nondifferentiability of F_B, we must take a closer look at the projection π_B.

We first define the faces of $B = \{z \mid \ell \le z \le u\}$. In this case, the faces are essentially determined by forcing some of the defining inequalities of B, namely $\ell \le z \le u$ to be satisfied as equalities. Thus, if I and J are disjoint subsets of $\{1, \ldots, n\}$, then a corresponding face of B is $\{z \in B \mid z_I = \ell_I, \;\; z_J = u_J\}$. For example, if $B = \mathbb{R}^n$, then B has only one face, namely B itself. On the other hand, if $B = \mathbb{R}^2_+$, the nonnegative orthant of $\mathbb{R}^2$, then the four faces of B are $(0,0)$, $0 \times \mathbb{R}_+$, $\mathbb{R}_+ \times 0$, and $\mathbb{R}^2_+$. These faces are critically related to π_B as we now show. Given a face F of the set B, let σ represent all the points in $\mathbb{R}^n$ that are projected onto F by π_B. The collection of all such σ is called the *normal manifold*. The sets σ are called cells of the normal manifold. It was shown in (Robinson 1992) that each cell is polyhedral and has dimension n and that this collection of cells forms a partition of $\mathbb{R}^n$. Returning to our two examples above, when $B = \mathbb{R}^n$, the only cell is $\sigma = \mathbb{R}^n$, which has dimension n and partitions $\mathbb{R}^n$. For $B = \mathbb{R}^2_+$, the four cells are the orthants of $\mathbb{R}^2$, each of which have dimension 2 and which partition $\mathbb{R}^2$.

The normal manifold in $\mathbb{R}^2$ corresponding to the box $B := [0, \infty) \times [0, 1]$ is given in Figure 2.

The projection $\pi_B(x)$ onto the box $B := [\ell, u]$ can be computed component-wise as follows

$$(\pi_B(x))_i = \begin{cases} \ell_i & \text{if } x_i < \ell_i, \\ x_i & \text{if } \ell_i \le x_i \le u_i, \\ u_i & \text{if } u_i < x_i. \end{cases} \tag{9}$$

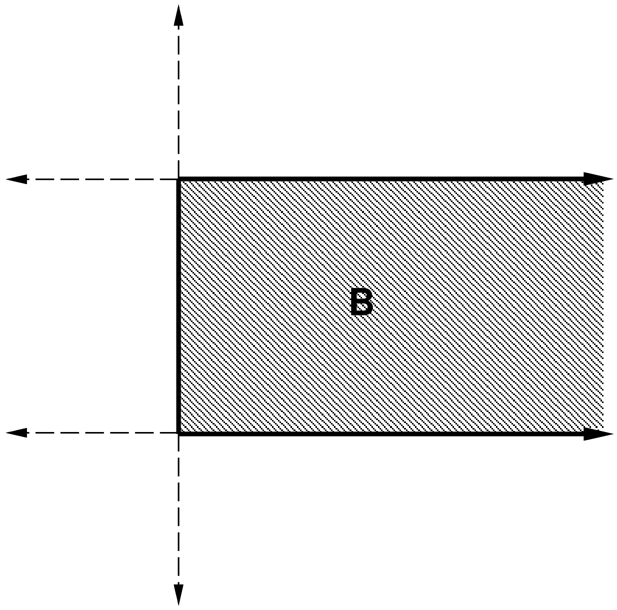

Figure 2: Normal Manifold for $B = [0, \infty) \times [0, 1]$

In this case, $\mathbb{R}^n$ is partitioned into at most 3^n rectangular cells where in each cell the function used to compute $\pi_B(x)_i$ is affine. Thus, the restriction of the projection operator π_B to each of these cells is affine.

The normal manifold provides a useful tool for working with the normal map, since it partitions $\mathbb{R}^n$ into a number of cells on each of which π_B is affine. This in turn allows us to view F_B as a smooth nonlinear function on the interior of each each of these cells, or as a piecewise-smooth function over the whole space.

In Newton's method for smooth functions, the function F is linearized around a current iterate x^k. In order to linearize F_B around x^k, we must linearize the function F around the point $\pi_B(x^k)$. The projection operator, being already piecewise-linear, is left as is. This linearization yields a piecewise-linear normal map A_k defined by

$$A_k(x) := M\pi_B(x) + q + x - \pi_B(x), \tag{10}$$

where

$$M := F'(\pi_B(x^k)) \qquad \text{and} \qquad q := F(\pi_B(x^k)) - M\pi_B(x^k).$$

Construction of the Newton point x_N^k (i.e. the zero of A_k) is where the majority of the work in this algorithm is performed. We describe here a pivotal method for computing this point, given first by Ralph (1994), which results in a piecewise linear path from the current point x^k to the Newton point. The path p^k is parametrized by a variable t which starts at 0 and increases to 1 so that $p^k(0) = x^k$ and $p^k(1) = x_N^k$. The values of $p^k(t)$ at intermediate points in the path are generated to satisfy

$$A_k(p^k(t)) = (1-t)r, \tag{11}$$

where $r = F_B(x^k)$. To calculate the intermediate values of $p^k(t)$ we introduce the triplet $(z(t), w(t), v(t))$ which is defined using

$$z(t) := \pi_B(p^k(t)), \tag{12}$$

$$w(t) := (z(t) - p^k(t))_+, \quad v(t) := (p^k(t) - z(t))_+. \tag{13}$$

It is easy to see that

$$p^k(t) = z(t) - w(t) + v(t).$$

Substituting these variables into (11) gives the following system:

$$\begin{aligned} \begin{bmatrix} M & -I & I & r \end{bmatrix} \begin{bmatrix} z \\ w \\ v \\ t \end{bmatrix} &= -q + r \\ l \leq z &\leq u \\ w, v &\geq 0 \\ 0 \leq t &\leq 1. \end{aligned} \tag{14}$$

The pivotal technique we employ guarantees that the complementarity conditions (12) and (13) are satisfied at every point on the path. The simple bounds on z determine which cell of the normal manifold we are currently in, and thus the direction of movement is determined by solving the first equation of (14). Essentially, we remain in the current cell by fixing some of the variables at their bounds – these are the nonbasic variables. The remaining variables, the so called basic variables, should then correspond to linearly independent columns of the matrix $\begin{bmatrix} M & -I & I \end{bmatrix}$. These columns make up what is normally called the basis matrix. The first equation then uniquely determines a direction in which to move, so that $(z(t), w(t), v(t))$ are updated. When some of the components of ℓ and u are finite, a step in this direction will in general reach a bound before t reaches 1. In this case, the boundary of the current cell is reached and the set of fixed variables has to be updated. One variable is added to the fixed set, and another is released from its bound, thus determining a pivot. In this manner, a sequence of pivot steps is performed (similar to those of Lemke's method), each of which increases t, and that together form the piecewise-linear path $p^k(t)$ from x^k to x^k_N. If the bounds on z are infinite, this is a simple task, since t moves from 0 to 1 in one pivot step, assuming M is invertible. This single pivot step corresponds exactly to the direction finding step in a smooth Newton method.

The crucial property of this path is that the norm of the approximation A_k decreases linearly in t on the path, i.e.

$$\left\| A_k(p^k(t)) \right\| = (1 - t)A_k(x^k). \tag{15}$$

This property, fundamental to Newton's method, is illustrated in Figure 3. In this figure, the function $F(x) := Mx + q$, while the boxes are defined as $\mathbb{R}^2$ and $[0, \infty) \times [0, 1]$, respectively.

The contour lines shown represent the Euclidean norm of the piecewise-linear functions F_B, while the paths to the Newton point for each mapping are given by the dashed lines. Note the change in F_B at the cell boundaries evident in Figure 3. Each linear segment of each path represents a direction taken to minimize the norm of the affine map corresponding to the current cell of the normal manifold.

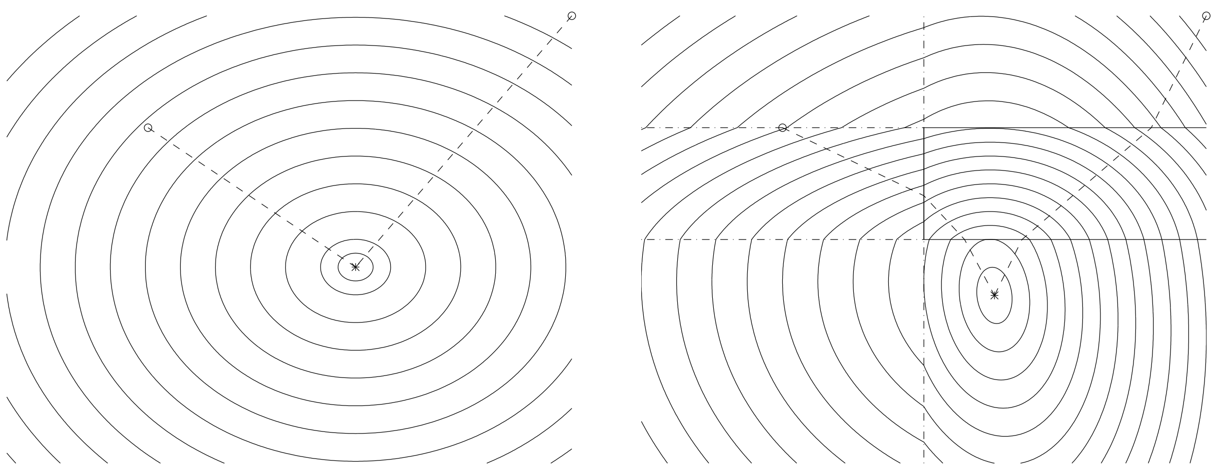

Figure 3: Contour Plots for F_B

A number of difficulties can arise while constructing the path. The initial basis may not be invertible, the value of t may not increase monotonically on the path, and the Newton point $p^k(1)$ may never be reached, due either to a cycling of bases or to ray termination. These issues are addressed in Section 4.

The path described above enables the application of damping techniques to achieve global convergence. In a pathsearch step, the path p^k is searched for a point $p^k(t)$ which satisfies

$$\left\| F_B(p^k(t)) \right\| \leq (1 - \sigma t) \left\| F_B(p^k(0)) \right\|, \tag{16}$$

where $\sigma \in (0, 1)$. If A_k is a "good" approximation to F_B at x^k, then (16) is satisfied for some $\bar{t} > 0$, since $\left\| A_k(p^k(t)) \right\|$ goes to zero linearly as t goes from 0 to 1. The new iterate x_{k+1} in the Newton process is defined to be $p^k(\bar{t})$. Given the appropriate hypotheses, it can be shown that the sequence $\left\| F_B(x^k) \right\|$ decreases linearly to 0 when $\bar{t}$ is chosen to satisfy (16).

In a typical linesearch or pathsearch method, the new iterate x^{k+1} is required to satisfy a descent condition similar to (16), which results in a monotonic decrease in $\|F_B\|$ or in a related merit function. There is evidence indicating that this requirement may impede or block convergence to the solution of the desired equation (Grippo, Lampariello & Lucidi 1986, Grippo, Lampariello & Lucidi 1991, Ferris & Lucidi 1994). Various non-monotone stabilization (NMS) schemes for Newton's method have been proposed, each seeking to improve efficiency by relaxing the requirement of monotone descent. The PATH solver implements a scheme of this type, modified to incorporate a pathsearch rather than a linesearch.

The NMS scheme implemented makes use two ideas. The first is an adaptation of the *watchdog* technique (Chamberlain, Powell & Lemarechal 1982) to reduce the number of pathsearches and therefore the number of function evaluations performed. The second is to allow a non-monotonic decrease in the merit function associated with the points chosen as a result of the pathsearches that are carried out. In the majority of cases, the watchdog technique elects to accept the Newton point if the point returned by the path generation procedure is suitably close to the current point. The measure of closeness, Δ, decreases as the algorithm progresses. In order to monitor these *d-steps*, the non-monotone descent condition (17) below is checked at least once every $\bar{n}$ iterations. The current merit function value is compared with a *reference value* $\mathcal{R}$, which is computed from previous function values. Steps in which these checks on the current merit function value occur are called *m-steps*. The points at which these criteria are checked *and satisfied* are called *check* points. An m-step is also taken when a d-step is unacceptable, that is, when it is too large. A watchdog step occurs when descent criteria are violated; when this occurs, the algorithm returns to the most recent check point, re-generates the path from the check point (if necessary), and searches the path for a point that satisfies

$$\left\| F_B(p^k(t)) \right\| \leq (1 - \sigma t)\mathcal{R}. \tag{17}$$

An example of how these different types of steps interleave is given in Section 5.

To summarize, the PATH solver uses a pivotal technique to construct a path $p^k(\cdot)$, parameterized by t, from the current point x^k to the Newton point x_N^k of the nonsmooth equation $F_B(x) = 0$. The watchdog stabilization technique is used to determine whether the path should be searched for a point $p^k(\bar{t})$ satisfying a nonmonotone descent condition, or if the Newton point should be accepted without searching the path. The algorithm spends most of its time in the path construction step, while the pathsearches, when they are performed, are also quite costly. How these steps are performed, and how the algorithm as a whole is implemented, is the topic of the next section.

4 Implementation Issues

The PATH solver is an implementation of the algorithm described in Section 3. It is written primarily in C, although it employs Fortran routines as well. Logically, it can be divided into two parts, the front end and solver. The front end acts as an interface to the routines which provide evaluations of the function F and its Jacobian J, the initial values and lower and upper bounds for the problems variables z, and other data specific to the problem. The solver is called by the front end to solve the MCP defined by F and B. The code for the solver remains the same, regardless of which front end is used. Compiler flags are used to ensure that calls to the proper front end are made.

The computational tests of the PATH solver have been done primarily using the GAMS front end and CPLIB, a library of routines used to evaluate F and J. These interface routines are written in Fortran for greater compatibility with the GAMS I/O library, a set of Fortran routines supplied by GAMS Development Corporation and required to interpret the problem as written to disk by the GAMS compiler.

Two other front ends exist as well. The first of these is a stand alone version, written in C and consisting of a main program whose task is simply to call the PATH solver and a number of auxiliary functions whose purpose is to evaluate F and J and provide the variable bounds, initial iterate, and other information required by the solver. Most recently, an AMPL link has been completed as well, which allows complementarity problems to be formulated in the AMPL modeling language (Fourer, Gay & Kernighan 1993) in a manner similar to GAMS. The AMPL complementarity link is programmed in C, as are the publicly available AMPL I/O routines (Gay 1993) necessary for interpreting the output of the AMPL compiler.

Although the PATH solver is written almost entirely in C, it is designed so that calls to numerical Fortran routines (e.g. for function evaluation) can be made without difficulty. Unfortunately, output to files opened in Fortran is not as easily accomplished in C, while the code to accomplish this would vary from platform to platform. The PATH solver avoids this problem by using macros to write to one of three well-defined files. When used with a Fortran front end, the macros expand to functions which write to strings and call Fortran routines to write the strings to files. When used with a C front end, the macros expand to functions which perform the desired output. This technique eliminates the need for separate output code throughout the solver; instead, one macro is used, whose definition depends on the front end in use.

At its top level, the solver consists of an outer iteration loop. At the beginning of each outer iteration, F and J are evaluated at the current point and a test for convergence is performed. Then, a linear subproblem is solved via a path construction technique and a new iterate obtained, perhaps as the result of a pathsearch. The time required to evaluate F and J varies from problem to problem and depends on the complexity and sparsity of F. If no pathsearch is necessary, the majority of the work in the outer loop is the construction of the path. When a pathsearch is performed, the work required depends on how quickly (17) can be satisfied.

The path is constructed by following a sequence of directions, each computed to lead to the zero of the affine map corresponding to the current cell. Each direction is followed until either the boundary of the current cell or the Newton point is reached. If a boundary is reached, the affine map changes, the direction needs to be recomputed, and we have a breakpoint in the path. Each successive affine map corresponds to a basis, each differing from its predecessor in only one column. Thus, the sequence of directions can be computed using any basis updating scheme. The PATH solver was tested using three different basis packages, the Harwell LA05 code (Reid 1982), the basis routines from MINOS 5.4 (Murtagh & Saunders 1983), and a simple basis package coded in C. We found that the MINOS routines were fastest, and have used those exclusively in subsequent testing, although the others can be substituted by setting appropriate compilation flags. The cell boundaries are found via a pivot step identical to that used in the simplex method for linear programming. We have chosen to employ a Devex-type choice of leaving variable in which the bounds are relaxed by a fixed slack tolerance (Harris 1973). This leads to fewer iterations and more stable bases.

We have assumed throughout that the basis corresponding to the current values of (z, w, v) is in fact invertible, although this is not always the case in practice. If the initial basis (as determined by x^k) is not invertible, the PATH solver cannot begin to construct the path. In this case, the current point is abandoned, and the PATH solver attempts to construct an invertible basis for system (14) by using as many slack columns as possible. If none of the z variables are free, this basis is always invertible. If all the variable are nonnegatively constrained, this slack basis is identical to that used to start Lemke's method for LCP, while the proper choice of the initial point $x = -w$ leads to a path which is identical to the one computed by Lemke's method. While this path cannot be searched to guarantee a decrease in $\|F_B\|$, it does provide an effective restart away from a singular basis. We refer to any basis containing the maximum number of slack columns as a Lemke basis.

As the path is constructed, the leaving variables determined by each pivot step are stored on a stack, so that the path can be reconstructed by using this information and the current basis. The path construction phase of the algorithm terminates at a Newton point, at the base of a ray, or as the result of an iteration interrupt. In the latter instances, the parameter t often fails to increase monotonically on the path. An option exists to truncate such paths so that their final point corresponds to a basis with the maximum t value attained on the path, since the norm of the approximation A_k is minimized at this point. In any case, the computed path is returned, in the form of a current basis and a stack, and the outer iteration loop continues.

Once the path is constructed, the stabilization techniques determine whether a pathsearch is necessary. If so, a backtrace of the path is performed, in which the path is reconstructed in reverse order from its computation, using the information stored on the stack. At each breakpoint, the descent condition (17) is checked. The search is terminated when (17) is satisfied. If no breakpoint satisfies these conditions, the initial segment of the path is searched using an Armijo technique (Armijo 1966). While backtracing the path is as expensive as the original construction, this is only necessary when a pathsearch is performed, and the number of searches is kept to a minimum by the watchdog stabilization technique. In addition, the path is backtraced only up the point satisfying (17), which may require little or no work at all. An obvious alternative to the backtrace is to search the path as it is constructed, but this leads to an excessive amount of essentially wasted function evaluations when the Newton point is ultimately chosen, and was found to be overly restrictive as well. Ideally, we would like to store the path in its entirety and save the expense of recomputing it, but this approach was rejected due to the large amount of memory this would require.

In the above descriptions of the algorithm and its implementation, a number of parameters and options, most unmentioned, exist and must be specified. For example, (17) depends on a tolerance σ, while the memory available to the basis routines is controlled by two "elbow room" tolerances. In addition, verbosity flags exist to print information useful to algorithm developer and user alike. The PATH solver provides default settings which are appropriate for most problems, but in order to speed solution for a difficult problem or to obtain detailed information, it may be necessary to modify some parameters. In all, there are 40 options which can be set via an options file, following the keyword–value syntax used by MINOS.

5 Computational Results

In this section, we compare the performance of the PATH solver to that of the undamped Newton method of Josephy (1979). Since Josephy's method can be viewed as a variant of our algorithm, (i.e. no pathsearch is carried out, and a particular choice of initial basis and basic values is made for each subproblem), a separate code for each method is not necessary. Instead, the Josephy method is the result of a particular setting of PATH solver options. This makes possible a meaningful comparison of solution times, as any differences are the result of the algorithm used and not of the implementation.

In order to compare the two methods, we have used them to solve the larger problems in a library of general equilibrium models provided by Tom Rutherford and formulated in the MPSGE language (Rutherford 1994*a*). The details of the CAMEROON model (GAMS model library sequence number 140) are given by Condon, Dahl & Devarajan (1987). The CO_2 model, number 142, is described by Perroni & Rutherford (1993) and is used to calculate the amount of global carbon emissions in the year 2020 under various scenarios. The ETAMACRO model of Manne (1977), number 144, is a macroeconomic interaction model for the United States, while the FINLAND model of Torma & Rutherford (1992), number 145, was developed to investigate the effects of tax reform in Finland. The GEMTAP model, number 146, is an updated version of a model for tax policy analysis described by Ballard, Fullerton, Shoven & Whalley (1984), while the VONTH model, number 156, is a land use model derived from MacKinnon (1976). Frequently in these model formulations, the first run is just for calibration purposes.

Table 2: PATH results

problem / run		major	minor	func	grad	time
CAMEROON		4	4	5	5	1.27
CO_2	2	13	51	14	14	3.48
	3	4	5	5	5	1.40
	4	4	5	5	5	1.23
	5	5	6	6	6	1.35
	6	6	10	7	7	1.80
	7	7	13	8	8	2.04
ETAMACRO		21	101	52	22	5.04
FINLAND	2	4	212	5	5	4.75
	3	4	4	5	5	3.07
	4	4	13	5	5	2.90
	5	4	4	5	5	3.11
GEMTAP	2	24	70	43	25	28.64
	3	6	8	7	7	6.50
	4	6	10	7	7	6.42
	5	26	134	70	27	33.90
VONTH		13	74	14	14	1.72

Table 2 indicates the number of major and minor iterations required to solve each problem using the PATH solver, along with the number of function and gradient evaluations and the amount of time required. Similar results obtained using Josephy-Newton's method are given in Table 3. The bar graph in Figure 4 demonstrates the difference in solution

Table 3: Josephy results

problem / run		major	minor	func	grad	time
CAMEROON		failed				
CO_2	2	failed				
	3	4	806	5	5	4.99
	4	4	810	5	5	4.92
	5	5	1002	6	6	6.52
	6	6	1240	7	7	7.67
	7	7	1436	8	8	10.08
ETAMACRO		failed				
FINLAND	2	4	838	5	5	8.31
	3	4	674	5	5	7.87
	4	4	821	5	5	9.00
	5	4	688	5	5	6.53
GEMTAP	2	failed				
	3	6	1993	7	7	30.90
	4	6	2003	7	7	30.20
	5	failed				
VONTH		13	979	14	14	3.95

times between the two algorithms by plotting the times required to solve the GEMTAP and FINLAND problems for each of the runs in which Josephy's method was able to compute a solution. These tables show that the PATH solver can be expected to solve general equilibrium problems in a robust manner, as opposed to the undamped method of Josephy. In addition, the PATH solver requires considerably less solution time, due to the smaller number of pivots performed. This is the result of the warm start taken by the PATH solver on the subproblems; in most cases, the optimal basis remains the same over the last few subproblems, so that only one pivot step is required for each.

Of particular interest is the ETAMACRO model formulated by Manne (1977) and modified to have shorter time periods of four years' duration. While Josephy's method diverges when applied to this problem, the stabilization techniques employed by the PATH solver result in a solution. Table 4 contains data from a run of the PATH solver on the modified ETAMACRO problem.

In Table 4, information from the log line printed at each major iteration is given. This demonstrates the behavior of the stabilization techniques used. The first two subproblems solved terminate at the Newton point. Note that for numerical reasons, the PATH solver forces t from 1 down to 0 instead of vice versa. Thus, the point $p^k(0)$ is the Newton point at iteration k. Note also that $\|F_B(\hat{x}^1)\|$ and $\|F_B(\hat{x}^2)\|$ are very much larger than $\|F_B(x^0)\|$. The algorithm parameters were set to check the descent condition after two iterations, so that instead of accepting $\hat{x}^2$, the algorithm returns to the check point x^0. In doing so, the points $\hat{x}^1$ and $\hat{x}^2$ are discarded. In the table, this is indicated by the first horizontal line. The path

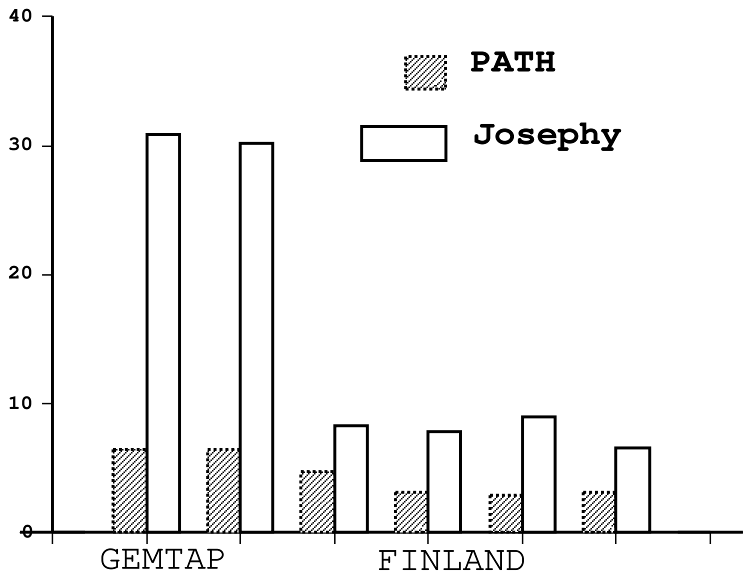

Figure 4: Comparison of Solution Times

Iterate	pivots	F evals	t value	$\|F_B\|$
x^0	0	1		5.1863e+01
$\hat{x}^1$	16	2	0	3.1458e+06
$\hat{x}^2$	12	3	0	6.0384e+06
x^0	0			5.1863e+01
x^1	16	21	.874	4.5335e+01
$\bar{x}^2$	12	22	0	3.1459e+06
$\bar{x}^3$	12	23	0	6.0384e+06
x^1	0			4.5335e+01
x^2	12	36	.596	3.4070e+01
x^3	2	37	0	3.8274e+01
x^4	1	38	0	1.5150e+01
x^5	1	39	0	6.0979e+00
x^6	1	40	0	1.7764e+00
x^7	1	41	0	2.5578e-01
x^8	1	42	0	7.0422e-03
x^9	1	43	0	5.6718e-06
x^{10}	1	44	0	3.6886e-12

Table 4: PATH output, modified ETAMACRO

is reconstructed from x^0, and instead of accepting the resulting Newton point, a backtracing pathsearch is performed. This pathsearch terminates at $t = .874$ and a point x^1 which becomes the new check point. The pathsearch required 18 function evaluations and resulted in a decrease in $\|F_B\|$. The algorithm continues from x^1 by taking two Newton steps $\bar{x}^2$ and $\bar{x}^3$, but again, the descent conditions are not satisfied. Thus, a watchdog step is performed and the algorithm returns to the check point x^1, as indicated by the second horizontal line in Table 4. The path from x^1 is reconstructed, and the nonmonotone linesearch procedure gives $t = .596$ and the new check point x^2.

The Newton point x^3 does not satisfy any monotone descent criterion; it is, however, accepted by the watchdog method. This is fortunate, since from this point on, the PATH solver computes Newton points which satisfy any reasonable descent criteria. Note that the optimal basis has been reached at this point, so that each succeeding iteration requires only one pivot step. Each of these single pivot steps result in a linear path from the current iterate to the Newton point.

Although the above problems are too small to demonstrate the efficiency of PATH for solving very large general equilibrium problems, it is noted elsewhere (Dirkse & Ferris 1994*a*) that PATH is an effective solver for large scale MCP's. For instance, an optimal control example with 8192 variables and constraints is solved in less that 1 hour on a DECstation 5000.

6 Conclusions

In this paper we have described an algorithm, PATH, for solving Mixed Complementarity Problems. The algorithm is shown to be markedly superior to the standard Newton method for such problems, both in speed and robustness. Combined with GAMS/MCP and MPSGE, PATH represents a valuable tool for computing general equilibria. We believe this advances the state of the art in this area.

References

Anstreicher, K. M., Lee, J. & Rutherford, T. F. (1992), 'Crashing a maximum-weight complementarity basis', *Mathematical Programming* **54**(3), 281–294.

Armijo, L. (1966), 'Minimization of functions having Lipschitz-continuous first partial derivatives', *Pacific Journal on Mathematics* **16**, 1–3.

Ballard, C., Fullerton, D., Shoven, J. & Whalley, J. (1984), *A General Equilibrium Model for Tax Policy Evaluation*, University of Chicago Press.

Chamberlain, R. M., Powell, M. J. D. & Lemarechal, C. (1982), 'The watchdog technique for forcing convergence in algorithms for constrained optimization', *Mathematical Programming Study* **16**, 1–17.

Colville, A. R. (1968), A comparative study on nonlinear programming codes, Technical Report 320–2949, IBM New York Scientific Center.

Condon, T., Dahl, H. & Devarajan, S. (1987), 'Implementing a computable general equilibrium model on GAMS – the Cameroon model', DRD Discussion Paper 290. The World Bank, Washington DC.

Dirkse, S. P. & Ferris, M. C. (1994*a*), MCPLIB: A collection of nonlinear mixed complementarity problems, Technical Report 1215, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin. Submitted for publication.

Dirkse, S. P. & Ferris, M. C. (1994*b*), 'The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems', *Optimization Methods & Software*. To appear.

Dirkse, S. P., Ferris, M. C., Preckel, P. V. & Rutherford, T. (1993), The GAMS callable program library for variational and complementarity solvers, Manuscript.

Ferris, M. C. & Lucidi, S. (1994), 'Nonmonotone stabilization methods for nonlinear equations', *Journal of Optimization Theory and Applications*.

Fourer, R., Gay, D. M. & Kernighan, B. W. (1993), *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, South San Francisco, CA.

Gay, D. M. (1993), 'Hooking your solver to AMPL', Numerical Analysis Manuscript 93–10. AT&T Bell Laboratories, Murray Hill, NJ.

Grippo, L., Lampariello, F. & Lucidi, S. (1986), 'A nonmonotone line search technique for Newton's method', *SIAM Journal of Numerical Analysis* **23**, 707–716.

Grippo, L., Lampariello, F. & Lucidi, S. (1991), 'A class of nonmonotone stabilization methods in unconstrained optimization', *Numerische Mathematik* **59**, 779–805.

Harker, P. T. & Pang, J.-S. (1990), 'Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms, and applications', *Mathematical Programming* **48**(2), 161–220.

Harris, P. M. J. (1973), 'Pivot selection methods of the Devex LP code', *Mathematical Programming* **5**, 1–28.

Hiriart-Urruty, J.-B. & Lamarechal, C. (1993), *Convex Analysis and Minimization Algorithms I*, Vol. 305 of *Grundlehren der mathematischen Wissenschaften*, Springer Verlag.

Josephy, N. H. (1979), Newton's method for generalized equations, Technical Summary Report 1965, Mathematics Research Center, University of Wisconsin–Madison, Madison, Wisconsin.

Lemke, C. E. & Howson, Jr., J. T. (1964), 'Equilibrium points of bimatrix games', *SIAM Journal of Applied Mathematics* **12**, 413–423.

MacKinnon, J. G. (1976), 'A technique for the solution of spatial equilibrium models', *Journal of Regional Science* **16**(3), 293–307.

Manne, A. S. (1977), ETA-Macro: A Model of energy-economy interactions, *in* C. J. Hitch, ed., 'Modeling Energy-Economy Interactions', Resources for the Future, Washington DC.

Mathiesen, L. (1987), 'An algorithm based on a sequence of linear complementarity problems applied to a Walrasian equilibrium model: An example', *Mathematical Programming* **37**, 1–18.

Murtagh, B. A. & Saunders, M. A. (1983), MINOS 5.0 user's guide, Technical Report SOL 83-20, Department of Operations Research, Stanford University, CA.

Ortega, J. M. & Rheinboldt, W. C. (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press.

Perroni, C. & Rutherford, T. (1993), 'International trade in carbon emission rights and basic materials: General equilibrium calculations for 2020', *Scandinavian Journal of Economics*.

Ralph, D. (1994), 'Global convergence of damped Newton's method for nonsmooth equations, via the path search', *Mathematics of Operations Research*. To appear.

Reid, J. K. (1982), 'A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases', *Mathematical Programming* **24**, 55–69.

Robinson, S. M. (1992), 'Normal maps induced by linear transformations', *Mathematics of Operations Research* **17**(3), 691–714.

Robinson, S. M. (1993), 'Newton's method for a class of nonsmooth functions', *Set Valued Analysis*. To appear.

Rutherford, T. F. (1994*a*), Applied general equilibrium modeling with MPSGE as a GAMS subsystem, Manuscript, Department of Economics, University of Colorado, Boulder.

Rutherford, T. F. (1994*b*), Extensions of GAMS for complementarity problems arising in applied economic analysis, Manuscript, Department of Economics, University of Colorado, Boulder.

Torma, H. & Rutherford, T. (1992), 'A general equilibrium assessment of Finland's grand tax reform', working paper 15/1992. Department of Economics and Management, University of Jyvaskyla, Finland.