

COMPUTING NASH EQUILIBRIA BY ITERATED POLYMATRIX APPROXIMATION

SRIHARI GOVINDAN AND ROBERT WILSON

ABSTRACT. This article develops a new algorithm for computing a Nash equilibrium of an N -person game. The algorithm approximates the game by a sequence of polymatrix games. We provide sufficient conditions for local convergence to an equilibrium and report on our computational experience. The algorithm converges rapidly but it is not failsafe. We show that if the algorithm does not converge then one can at any point switch easily to the Global Newton Method, which is slower but failsafe. Thus, the algorithm can be used to obtain a fast start for the Global Newton Method.

1. INTRODUCTION

The standard algorithms for calculating Nash equilibria of N -player games use homotopy methods to trace equilibria in the graph of the Nash correspondence. The equilibria are traced above a path of games from a starting point that is a game whose unique equilibrium is known, to the terminal point that is the target game whose equilibria are wanted (Eaves, 1972; Eaves and Schmedders, 1999). In some versions the homotopy is implicit because calculations are conducted in the strategy space, as in the adaptations of the basic Lemke and Howson (1964) algorithm by Lemke (1965), Shapley (1974), and Wilson (1992) for 2-player games and its extensions by Rosenmüller (1971) and Wilson (1971) for N -player games. In some implementations the homotopy is traced explicitly on the vertices of the sub-simplices in a pseudo-manifold, as in Eaves (1972, 1984), Eaves and Scarf (1981), Scarf and Hansen (1973), and the version implemented in the package *Gambit* by McKelvey, McLennan, and Turocy (1996); cf. McKelvey and McLennan (1996). The principal exceptions to the standard reliance on homotopy methods are the approximation via quantal-response equilibria developed by McKelvey and Palfrey (1995), and the interior-point method developed by van den Elzen and Talman (1991).

In a previous article (Govindan and Wilson, 2001) we describe an application of Smale's (1976) Global Newton Method to computing equilibria of N -player games. We show that its path is a homotopy whose image in the strategy space coincides when $N = 2$ with the path of the Lemke-Howson algorithm. For general $N \geq 2$, it finds a sequence of equilibria whose indices alternate between $+1$ and -1 if the game is generic. Our purpose in this article is to improve the performance of the Global Newton Method by providing a "fast start" that jumps quickly to the vicinity of the target game. The next paragraphs suggest the motivation for our approach.

1.1. Motivation. For games with more than two players, homotopy methods follow paths in the strategy space that are nonlinear (actually, piecewise analytic). Traversing nonlinear paths requires many small steps, or larger steps require error-correction procedures, or pseudo-manifold representations require successive refinements to obtain reasonable accuracy. The computational burden is compounded severely, moreover,

Date: March 29, 2001.

This work was funded in part by grants from the Social Sciences and Humanities Research Council of Canada and the National Science Foundation of the United States [SBR9511209].

by the characteristic feature that the path from the starting game to the target game has many twists and reversals. That is, the path typically involves many changes in the support of the strategies, and further, the trajectory frequently reverses orientation. A generic game along the path can have a number of equilibria that is exponential in the number of strategies (von Stengel, 1999), and the path can wind back and forth through most of these equilibria before making any further progress toward the target game. Even for games with only five players and five pure strategies per player, it is not unusual to see homotopy-based algorithms stalled for many iterations at each of a long series of games over each of which the trajectory oscillates back and forth through many equilibria (say, 30 or 40) with alternating indices $+1$ and -1 before resuming forward progress. All of this is consistent with Kohlberg and Mertens' (1986, Theorem 1) structure theorem, which says that the Nash graph is homeomorphic to the space of games, but it is discouraging to discover that the number of convolutions of the graph above a typical game—not the target game, just an intermediate game on the homotopy's path to reach it—can be exponentially large, and not just in theory but often in practice.

To improve computational efficiency, one needs an initial procedure that reaches quickly the vicinity of the target game. That is, if one can leap quickly to a pair (\tilde{G}, σ) in the Nash graph, where \tilde{G} is a game near the target game G and σ is one of its equilibria with index $+1$, then from the starting point (\tilde{G}, σ) the Global Newton Method can be applied efficiently to find all equilibria of the target game G accessible via the homotopy along the line segment from \tilde{G} through G that continues from (\tilde{G}, σ) . This is much quicker than tracing all the equilibria above the path from the usual starting point, which is a game sufficiently far on the ray from G that it has a unique equilibrium. After this fast start, the burden of computing all equilibria of G accessible via the ray as the Global Newton Method (or any other algorithm based explicitly or implicitly on a homotopy) winds its way through the convolutions of the graph above the target game is worth the effort because one is finding equilibria relevant for studies of the target game.

In this article we describe an algorithm that typically converges quickly to an equilibrium of a game near or at the target game. We also describe the pseudo-code for its implementation and report computational experience with a prototype. Its speed derives from three features. First, the method consists of iterative approximation of the target game, so right from the start it obtains equilibria of games close (actually, as elaborated in Section 2, tangent) to the target game. Second, the target game is approximated by a polymatrix game (Howson and Rosenthal, 1974), so the linear operations of the Lemke-Howson algorithm can be applied to obtain its accessible equilibria, and in particular to find its equilibrium with index $+1$ that is the first one accessible via a designated ray. The third aspect is a huge reduction in dimensionality: the Lemke-Howson algorithm operates on an $M \times M$ matrix, where $M = \sum_n m_n$ and m_n is the number of player n 's pure strategies. The combination of these three features greatly reduces the computations required to reach a point (\tilde{G}, σ) in the Nash graph with \tilde{G} close to G .

The reduction in dimensionality has implications for efficiently computing equilibria of games in extensive form. If the target game's normal form is generic then it has $N \times \prod_n m_n$ independently specified payoffs. For example, if the number of players is $N = 5$ and each $m_n = 5$, then the number of normal-form payoffs is $N \times \prod_n m_n = 15625$, but $M \times M = 625$. Thus, to compute players' expected payoffs from the normal form one must multiply each of the 15625 payoffs by the product of 4 probabilities (62500 multiplications if redundant products are not avoided). By exploiting the structure of the extensive form one can reduce considerably the computations required to compute the $M \times M$ matrix required by the approximation as a polymatrix game, and given this matrix, M^2 multiplications suffice to evaluate players' expected payoffs. For

instance, in a Centipede game in which player n has m_n opportunities to terminate the game, the extensive form has only NM distinct payoffs and construction of the Jacobian requires less than M^3 multiplications.

In fact, the lower dimensionality of the space of payoffs for a polymatrix game is an extreme case of the lower dimensionality for extensive form games, since a polymatrix game is one in which each player interacts bilaterally with each other player and each player's payoffs are additive over the bilateral interactions with other players (as in a "round robin" tournament). Typically, much of the special structure of the extensive form is inherited by its polymatrix approximations, and a subroutine that exploits this structure can greatly reduce storage requirements and the number of multiplications required. In this article, however, we address only the case of games specified in normal form.

1.2. Prior Literature. Our approach follows a suggestion by van den Elzen and Talman (1994). They propose to linearly approximate the payoff function for an N -player game by the first terms of a Taylor series expansion at a strategy $\hat{\sigma}$. This approximation is used to linearize the conditions for a Nash equilibrium. Their procedure uses an adaptation of the interior-point method in van den Elzen and Talman (1991) to solve the linearized conditions to obtain a new strategy σ . The process is then repeated using σ as the point at which the Taylor series is computed next. Although they report numerical results for three 3-player 2-strategy examples, our experience is that this procedure is rarely successful. Our formulation improves the operational performance of this kind of algorithm, demonstrates its local convergence properties, and shows that where it stalls the Global Newton Method can continue to a solution.

* * *

Section 2 sets up the notation and definitions and Section 3 describes the algorithm. Section 3.3 elaborates how the algorithm can stall and thus require continuation via the Global Newton Method to pass through singularities. Section 4 establishes conditions sufficient for a positive radius of convergence. Section 5 presents the pseudo-code and numerical results obtained from the prototype listed in the APL language in Appendix A. We refer the reader to our previous article (Govindan and Wilson, 2001) on the Global Newton Method for the pseudo-code and APL program for an implementation that finds all equilibria of the target game that are accessible via a homotopy along a generic ray emanating from the target game.

2. FORMULATION

Denote the set of players by $\mathcal{N} = \{1, \dots, N\}$, where $N > 1$. Use S_n to denote player n 's finite set of pure strategies, and Σ_n , player n 's simplex of mixed strategies. Let $S = \prod_n S_n$, $\Sigma = \prod_n \Sigma_n$, and $M = \sum_n m_n$ where $m_n = |S_n|$. The space Γ of payoffs for normal-form games with player set \mathcal{N} and pure strategy set S is the Euclidean space of dimension $N \times \prod_n m_n$.

For a game in Γ whose normal form is the $(N + 1)$ -dimensional array $A = ((A_s^n)_{s \in S})_{n \in \mathcal{N}}$ of payoffs from pure strategies, the expected payoff to player n from the pure strategy $s_n \in S_n$ when other players $n' \neq n$ use their mixed strategies in $\sigma \in \Sigma$ is the multilinear polynomial

$$G_{s_n}^n(\sigma; A) = \sum_{t \in S(-n)} A_{s_n, t}^n \prod_{n' \neq n} \sigma_{t(n')}$$

where $S(-n) = \prod_{n' \neq n} S_{n'}$. The M -vector of all payoffs is $G(\sigma; A) = ((G_{s_n}^n(\sigma; A))_{s_n \in S_n})_{n \in \mathcal{N}}$. A mixed strategy $\sigma \in \Sigma$ is an equilibrium of the game A if, for each $s \in S_n$ and $n \in \mathcal{N}$,

$$\sigma_s^n > 0 \quad \text{only if} \quad G_s^n(\sigma; A) = \max_{s' \in S_n} G_{s'}^n(\sigma; A).$$

A polymatrix game (or a bimatrix game if $N = 2$) is defined by the payoff $B_{s,s'}^n$ to player n from each pure strategy $s \in S_n$ and each pure strategy $s' \in S_{n'}$ of each other player $n' \neq n$, and n 's payoffs are additive over the bilateral interactions with other players. Thus, the space $\bar{\Gamma}$ of payoffs for polymatrix games is the Euclidean space of dimension $\sum_n m_n \sum_{n' \neq n} m_{n'}$. The M -vector of all payoffs is therefore $G(\sigma; B) = ((G_s^n(\sigma; B))_{s \in S_n})_{n \in \mathcal{N}}$, where

$$G_s^n(\sigma; B) = \sum_{n' \neq n} \sum_{s' \in S_{n'}} B_{s,s'}^n \sigma_{s'}^{n'}.$$

The linearity of the payoff function for a polymatrix game enables one to use a variant of the Lemke-Howson algorithm to calculate equilibria. The smaller dimensionality of polymatrix games, and the remarkable speed of the Lemke-Howson algorithm, are two motivations for using iterative approximation via polymatrix games.

A third motivation is the natural relationship between a general game $A \in \Gamma$ and a family of approximating polymatrix games in $\bar{\Gamma}$. In fact, one can consider this family to be the collection of polymatrix games corresponding to the hyperplanes tangent to the payoff function $G(\cdot; A)$ at various mixed strategies $\sigma \in \Sigma$. The Jacobian $\nabla G(\sigma; A)$ of the partial derivatives of the payoff function $G(\cdot; A) : \Sigma \rightarrow \mathbb{R}^M$ at $\sigma \in \Sigma$ is an $M \times M$ matrix with $m_n \times m_n$ -dimensional blocks of zeros along its diagonal. This Jacobian matrix is also the Jacobian matrix of the polymatrix game $B \in \bar{\Gamma}$ for which

$$B_{s,s'}^n = \partial G_s^n(\sigma; A) / \partial \sigma_{s'}^{n'}$$

for $s' \in S_{n'}$ and $n' \neq n$. Hereafter we say that the polymatrix game B approximates the game A at σ if their Jacobians agree at σ ; that is, $\nabla G(\sigma; B)$ is proportional to $\nabla G(\sigma; A)$.

The polymatrix game B that approximates A at σ has a natural interpretation: player n 's bilateral interaction with each other player $n' \neq n$ is approximated by averaging over the strategies of all other players $n'' \neq n, n'$, using their mixed strategies $\sigma^{n''}$ as the weights for computing the average payoff obtained from each combination of the pure strategies of players n and n' . Thus,

$$B_{s,s'}^n = \sum_{t \in S(-n,n')} A_{s,s',t}^n \prod_{n'' \neq n,n'} \sigma_{t(n'')}^{n''}.$$

Hereafter we use $DG_A(\sigma) \equiv \frac{1}{N-1} \nabla G(\sigma; A)$ so that $G(\sigma; A) = DG_A(\sigma) \cdot \sigma$ [see Lemma 3.1 below]. We can then interpret $DG_A(\sigma)$ as the Jacobian of the polymatrix game in $\bar{\Gamma}$ that approximates the payoff of $A \in \Gamma$ at σ . Similarly, we represent a polymatrix game $B \in \bar{\Gamma}$ by the constant Jacobian $DG_B = \nabla G(\sigma; B)$ of its payoff function.

Given the payoff array $A \in \Gamma$ for a game, and any vector $g \in \mathbb{R}^M$, define the perturbed game $A \oplus g \in \Gamma$ via $(A \oplus g)_{s,t}^n = A_{s,t}^n + g_s$ for each $t \in S(-n)$. Similarly, when B is a polymatrix game, define the perturbed polymatrix game $B \oplus g \in \bar{\Gamma}$ via $(B \oplus g)_{s,s'}^n = B_{s,s'}^n + g_s / (N-1)$ for each strategy $s' \in S_{n'}$ of another player $n' \neq n$. Note that when J is the Jacobian matrix DG_B of a polymatrix game $B \in \bar{\Gamma}$, the corresponding Jacobian of $B \oplus g$ is $J' \equiv J \oplus g$, where $J'_{s,s'} = J_{s,s'} + g_s / (N-1)$ for pure strategies s and s' of different players.

As in Kohlberg and Mertens (1986), it is sometimes convenient to use the representation $A = \tilde{A} \oplus g(A)$ where $g(A) = G(\tilde{\sigma}; A)$ using the mixed strategy $\tilde{\sigma}$ that assigns equal probability $1/m_n$ to each pure strategy of player n . The payoff array \tilde{A} is obtained from A by subtracting $g(A)_s^n/|S(-n)|$ from each payoff $A_{s,t}^n$, $t \in S(-n)$. Similarly, for a polymatrix game, $B = \tilde{B} \oplus g(B)$ where $g(B) = G(\tilde{\sigma}; B)$.

3. ITERATIVE APPROXIMATION

The algorithm is designed to exploit the following property.

Lemma 3.1. *A mixed strategy $\sigma \in \Sigma$ is an equilibrium of the game $A \in \Gamma$ if and only if it is an equilibrium of the polymatrix game $B \in \bar{\Gamma}$ that approximates A at σ , i.e., $DG_B = DG_A(\sigma)$.*

Proof. Because the payoff function $G(\cdot; A)$ is a polynomial that is homogeneous of degree $N - 1$, Euler's Theorem implies that $G(\sigma; A) = DG_A(\sigma) \cdot \sigma$, where $DG_A(\sigma) \equiv \frac{1}{N-1} \nabla G(\sigma; A)$ as defined previously. Consequently, the defining property of an equilibrium that

$$\sigma_s^n > 0 \quad \text{only if} \quad G_s^n(\sigma; A) = \max_{s' \in S_n} G_{s'}^n(\sigma; A)$$

is equivalent when $J = DG_B = DG_A(\sigma)$ to

$$\sigma_s^n > 0 \quad \text{only if} \quad J_s^n \cdot \sigma = \max_{s' \in S_n} J_{s'}^n \cdot \sigma,$$

which is the definition that σ is an equilibrium of the polymatrix game B whose Jacobian is J . □

An immediate corollary is that if σ is a regular equilibrium of A , then its index (+1 or -1) agrees with the index of σ as an equilibrium of B , since in either case the index is the sign of the determinant of a Jacobian matrix derived from DG_B , as in Güel, Pearce, and Stacchetti (1993).

Note that if one uses the representation $A = \tilde{A} \oplus g$ then Lemma 3.1 says that σ is an equilibrium of A iff it is an equilibrium $\tilde{B} \oplus g$ where $DG_{\tilde{B}} = DG_{\tilde{A}}(\sigma)$.

To solve a polymatrix game $B \in \bar{\Gamma}$, we use the variant of the Lemke-Howson algorithm described in Govindan and Wilson (2001). Choose a generic ray $g \in \mathbb{R}^M$ and a scalar $\lambda^* > 0$ sufficiently large that the polymatrix game $B \oplus \lambda^* g$ has the unique pure strategy equilibrium for which $\sigma_s^n = 1$ for $s = \text{Arg max}_{s' \in S_n} g_{s'}$. Then trace the 1-dimensional manifold of equilibria $\sigma(\lambda)$ of the games $\{B \oplus \lambda g | \lambda \in [0, \lambda^*]\}$ parameterized by λ as $\lambda \downarrow 0$. Tracing the equilibria $\sigma(\lambda)$ is a linear operation consisting of a series of pivots (Gaussian eliminations) that continues until $\lambda = 0$. Similarly, if $B = \tilde{B} \oplus g$ and the algorithm is applied to \tilde{B} using the same ray g , then one stops at $\lambda = 1$. By continuing the homotopy beyond the first terminus one can compute all equilibria accessible via the ray g . For instance, if the graph of equilibria is S-shaped above the line of games $\{B \oplus \lambda g\}$ in a neighborhood of $\lambda = 0$ then the homotopy finds all three equilibria and their indices are +1, -1, and +1 in the sequence of crossings of $\lambda = 0$. This algorithm is very fast; e.g., in our implementation in APL on a Pentium III computer, a polymatrix game with 5 players, each with 6 pure strategies, requires an average of 0.03 seconds to reach the first equilibrium of the target game B and the other accessible equilibria are found quickly after that. In contrast, the Global Newton Method requires about 1000 seconds to reach the first equilibrium of the general game A that B approximates.

3.1. The Iterative Polymatrix Approximation Algorithm. A naive version of the IPA algorithm for a general game $A \in \Gamma$ proceeds by iteratively improving the strategy at which the approximation is computed, as in the following procedure:

1. Start by specifying a mixed strategy $\hat{\sigma}$ at which the game A is approximated initially. Also, use a generic vector $g \in \mathbb{R}^M$ to define the ray for the homotopy used by the Lemke-Howson algorithm.
2. Approximate A by the polymatrix game B for which $DG_B = DG_A(\hat{\sigma})$.
3. Use the Lemke-Howson algorithm to find the first equilibrium σ of B reached by the homotopy that traces equilibria along the path of games $\{B \oplus \lambda g\}$ as $\lambda \downarrow 0$.
4. If σ is sufficiently close to $\hat{\sigma}$ then either terminate with an approximate equilibrium, or invoke the Global Newton Method to find all equilibria of A accessible via the induced ray $g(\sigma)$ [defined below].
5. Otherwise, move $\hat{\sigma}$ “toward” σ and return to Step 2.

The merits of this procedure lie in Steps 3 and 4. As mentioned, Step 3 is very fast. Step 4 reflects the fact that the procedure can be interrupted to allow continuation via the Global Newton Method, which is slower but failsafe. Continuation is possible for two reasons. First, the Lemke-Howson algorithm produces an equilibrium σ of a game $A \oplus g(\sigma)$ near A with the index $+1$, which is a key property required. To see this, recall that for each λ on the path of the homotopy, the equilibrium $\sigma(\lambda)$ obtained by the Lemke-Howson method is a fixed point of the map $f : \Sigma \rightarrow \Sigma$ for which $f(x) = r(x + [DG_B \oplus \lambda g] \cdot x)$, where $r : \mathbb{R}^M \rightarrow \Sigma$ is the retraction defined by Gül, Pearce, and Stacchetti (1993). Because $\sigma = \sigma(0)$ is the first equilibrium of B on the trajectory as $\lambda \downarrow 0$, its index is $+1$ (and depends only on the Jacobian). Second, using Lemma 3.1, σ is also an equilibrium of the game $A \oplus g(\sigma) \in \Gamma$ for which $g(\sigma) = [DG_B - DG_A(\sigma)] \cdot \sigma$. Therefore, the Global Newton Method can be continued from the starting point $(A \oplus g(\sigma), \sigma)$ along the line $\{A \oplus \lambda g(\sigma) | 0 \leq \lambda \leq 1\}$ as again λ decreases from 1 to 0 to find all equilibria of A accessible via the ray $g(\sigma)$.

There are two major deficiencies of the naive version above. The first is that Step 5 requires a sophisticated improvement procedure, and the second is that Step 4 may require continuation via the Global Newton Method because the procedure can cycle or stall. Each of these is addressed below, and then in Section 4 we present sufficient conditions for convergence.

3.2. An Improvement Procedure. The main difficulty in Step 5 is that the Jacobian J of the map $\hat{\sigma} \mapsto \sigma$ is difficult to compute. (If it were easy to compute then one could apply directly a local Newton method or the Global Newton Method.) Our implementation has obtained good results with the simple expedient of using the Rule of False Position. The off-diagonal entries of the Jacobian are taken to be zero, and each entry J_{ss} on the diagonal is approximated by the rate of change of the displacement $\sigma_s - \hat{\sigma}_s$ as a function only of $\hat{\sigma}_s$, as measured over the previous two iterations of Step 3. This is equivalent to a quasi-Newton method based on a local approximation of the Jacobian of the map $\hat{\sigma} \mapsto \sigma$. Thus, from Step 5 one could return to Step 2 with the new mixed strategy $\hat{\sigma}$ used for the approximation being the value of $\hat{\sigma} - \alpha J^{-1} \cdot [\sigma - \hat{\sigma}]$ computed in Step 5, where J is the diagonal matrix and $\alpha \in (0, 1]$ is a small stepsize parameter. This is the improvement procedure implied by the local quasi-Newton method.

In fact, however, the restriction to the strategy space Σ complicates matters so it is better to apply this same procedure to values of the transformed variable $z \in \mathbb{R}^M$ defined by $z \equiv \sigma + DG_B \cdot \sigma$ and then to recover $\sigma = r(z)$ by using the retraction r defined by Gül, Pearce, and Stacchetti (1993). That is, $r(z)$ is the unique value of $\sigma \in \Sigma$ for which $[\sigma' - \sigma] \cdot [z - \sigma] \leq 0$ for all $\sigma' \in \Sigma$. Computing the retraction is easy

and almost instantaneous, and experience has shown that working in the full space \mathbb{R}^M of z -values is more robust. Thus, Step 1 begins with an initial choice \hat{z} , Step 2 uses $\hat{\sigma} = r(\hat{z})$, and Step 5 returns to Step 2 with $\hat{z} - \alpha J^{-1} \cdot [z - \hat{z}]$, where $z = \sigma + DG_B \cdot \sigma$ using σ from Step 3, and the diagonal of J measures the rate of change of z as a function of \hat{z} over the previous two iterations.

Although there are techniques for estimating the full Jacobian of the map $\hat{z} \mapsto z$ using data from multiple previous iterations, we have not implemented these and our numerical results reflect only the simple procedure of estimating the diagonal from the previous two iterations.

3.3. Cycling. In Section 4 we establish sufficient conditions for local convergence of local Newton methods. However, it is fundamental that the IPA algorithm can stall—which is why we emphasize the importance of the ability to continue from any intermediate point using the Global Newton Method. The reason a stall occurs is that the trajectory $\sigma(\hat{\sigma})$, interpreted as a function of the strategy $\hat{\sigma}$ used to obtain the polymatrix approximation B for which $DG_B = DG_A(\hat{\sigma})$, can encounter a “membrane” (a subset of codimension 1) on either side of which Step 4 implies movement toward the membrane. Thus the IPA algorithm cycles in the vicinity of such a membrane. A membrane of codimension 1 is no impediment to the version of the Global Newton Method due to Keenan (1981) because its trajectory passes through singularities of codimension 1, but this is not true of the IPA algorithm, as explained in Govindan and Wilson (2001).

To illustrate, suppose the equilibrium graph for the polymatrix approximation $B \oplus \lambda g$ at $\hat{\sigma}$ is S-shaped above a neighborhood of $\lambda = 0$: then Step 5’s movement of $\hat{\sigma}$ toward the first equilibrium σ of B , say at the top of the S, can move the graph sufficiently that the top two equilibria disappear, and then the first equilibrium of the next polymatrix approximation is at the bottom of the S—and a further movement in that direction makes the first equilibrium reappear at the top of the S. As in the Global Newton Method, such cycles might be avoided by reversing orientation: upon reaching a membrane, Step 5 improves σ by retreating away from (not toward) σ and continuing on the branch of the S with index -1 , namely, by choosing σ in Step 3 to be the *second* equilibrium reached by the Lemke-Howson algorithm. However, we have not yet developed the full implications of this approach because of the apparent advantages of moving out of a stall by using the Global Newton Method—and our main purpose here is to present an algorithm that enables a fast start for the Global Newton Method.

4. RADIUS OF CONVERGENCE

In this Section we provide a sufficient condition for our algorithm to converge locally to an equilibrium. More specifically, we construct a map whose fixed points, if they exist, are locally stable equilibria under the algorithm.

We say that a subset of games is generic if its complement in Γ has lower dimension, and we say that a game with payoff array A is generic if it lies in a generic subset of Γ . Because $DG_A(\cdot)$ is a polynomial, and equilibria are defined by polynomial equalities and inequalities, all the sets and functions we deal with here are semi-algebraic.

Let \bar{E} be the graph of the equilibrium correspondence over the domain $\bar{\Gamma}$ of polymatrix games, and let $proj: \bar{E} \rightarrow \bar{\Gamma}$ be the natural projection. Also let \mathcal{S} be the unit sphere in \mathbb{R}^M . According to Govindan and Wilson (2001), for each polymatrix game B there exists a lower-dimensional semi-algebraic subset $C(B) \subset \mathcal{S}$ of critical points such that for each $g \in \mathcal{S} \setminus C(B)$ the subset $proj^{-1}(\{B \oplus \lambda g | \lambda > 0\})$ of \bar{E} is a 1-dimensional

manifold on which, for all but a finite number of values of λ , all the equilibria of $B \oplus \lambda g$ are regular (i.e., isolated, each with index +1 or -1).

Fix a game $A \in \Gamma$ such that all its equilibria are regular (which is a property of generic games). Let $X = \{(\sigma, g) \in \Sigma \times \mathcal{S} \mid g \in C(DG_A(\sigma))\}$ be the set of critical pairs. Then, applying the Generic Local Triviality Theorem (cf. Bochnak, Roy, and Coste, 1998) to the projection map from X to Σ , we have that $\dim(X) < (M - N) + (M - 1)$. Once again by the Generic Local Triviality Theorem (applied now to the projection map from X to \mathcal{S}) we have that for generic g the dimension of $\{(\sigma, g) \in X\}$ is less than $M - N$. Therefore, for each g in a generic subset of \mathcal{S} the dimension of $\{\sigma \mid (\sigma, g) \in X\}$ is also less than $M - N$. Since the set $E(A)$ of equilibria of the game A is finite, it also true that for generic $g \in \mathcal{S}$, $(\sigma, g) \notin X$ if $\sigma \in E(A)$.

To sum up so far, for each generic $g \in \mathcal{S}$, there exists a semi-algebraic, open and dense subset Σ^* of Σ such that for each $\sigma \in \Sigma^*$ the equilibrium correspondence over $\{DG_A(\sigma) \oplus \lambda g \mid \lambda > 0\}$ is a 1-dimensional manifold. Moreover, $E(A) \subset \Sigma^*$. Now fix a generic $g \in \mathcal{S}$, and let $f : \Sigma^* \rightarrow \Sigma$ be the function that assigns to each $\sigma \in \Sigma$ the first equilibrium of $DG_A(\sigma)$ on the manifold that is reached as λ decreases from ∞ to 0. By construction, a fixed point of f is an equilibrium with index +1 of the game A . However, in as much as f might not be extendable to a continuous function over Σ , it might not have a fixed point. It can be shown that if the complement of Σ^* has codimension 2 or more, then f has a fixed point. But this condition, while satisfied by an open set of games, is not generic.

Let $Z^* = r^{-1}(\Sigma^*)$, where r is the retraction map defined in Section 3. Define $F : Z^* \rightarrow \mathbb{R}^M$ by $F(z) = f(r(z)) + G(f(r(z)); B)$ where $B = DG_A(r(z))$. Since F is a semi-algebraic function on an open semi-algebraic subset, it is differentiable (in fact, analytic) almost everywhere. Let $D : Z^* \rightarrow \mathbb{R}^M$ be the corresponding displacement map; i.e., $D(z) = z - F(z)$.

Theorem 4.1. *If F has a fixed point z then there is a neighborhood U of z such that starting from any point in U the Newton method applied to the displacement map D converges to z .*

Proof. Let z^* be a fixed point of F . Then $\sigma^* = r(z^*)$ is an equilibrium of both the polymatrix game $B^* = DG_A(\sigma^*) \in \bar{\Gamma}$ and the game $A \in \Gamma$, and it is regular (since A is a generic game) with a positive index. Therefore, all polymatrix games close to B^* have a unique equilibrium with +1 index close to σ^* , which is reachable by the unique path starting at infinity over the ray through g . For any point z in a neighbourhood of z^* , the Newton field for the game A , call it $v(\cdot)$, points in the direction of the unique equilibrium of the associated polymatrix game B that is close to z^* . But, that is precisely the vector field $-D$ as well. Since the index of z^* under the vector field $-v$ is +1, its index under D is also +1. The result now follows from the fact that a regular +1 zero is always locally stable under the Newton method. \square

Although the Theorem ensures local convergence of the Newton method, two practical issues remain. One is whether it suffices to rely on a quasi-Newton method that uses only an approximation of the diagonal of the Jacobian. The other is whether typically the algorithm is globally convergent, so that only occasionally must one rely on the Global Newton Method to pass through a stall. The numerical results in the next section seem to answer these questions affirmatively.

5. PSEUDOCODE AND NUMERICAL RESULTS

The following pseudocode includes the amendments discussed in Section 3. We also add here two optional features in steps 4 and 7 below to speed calculations.

1. Select a stepsize $\alpha \in (0, 1)$ and a generic vector $g \in \mathbb{R}^M$. Start with an initial $\hat{z} \in \mathbb{R}^M$.
2. Compute $\hat{\sigma} = r(\hat{z})$.
3. Compute $DG_A(\hat{\sigma})$, interpreted as the Jacobian of the polymatrix game B that approximates A at $\hat{\sigma}$.
4. If the solution for σ remains optimal using the support of σ in the previous iteration then skip to Step 6.
5. Use the Lemke-Howson algorithm to find the equilibrium σ of B with index +1 that is the first equilibrium of B reached by the homotopy that traces equilibria along the path of games $\{B \oplus \lambda g\}$ as $\lambda \downarrow 0$.
6. Set $z = \sigma + DG_A(\hat{\sigma}) \cdot \sigma$.
7. If the angle between $\hat{z} - \hat{\sigma}$ and $z - \hat{\sigma}$ is acute then rescale $\hat{z} - \hat{\sigma}$ to match $z - \hat{\sigma}$, which is innocuous because such rescaling has no effect on the retraction in Step 2.
8. If $\|z - \hat{z}\|$ is sufficiently small then $\sigma \approx \hat{\sigma}$ so exit with an approximate equilibrium of A .
9. If this is the first iteration then compute the improved estimate \hat{z} that is $\tilde{z} = \hat{z} + \alpha[z - \hat{z}]$ and proceed to Step 11.
10. Use the Rule of False Position to improve the estimate \hat{z} ; that is, for each $s \in S_n$, $n \in \mathcal{N}$, compute

$$\tilde{z}_s^n = [1 - \alpha]\hat{z}_s^n + \alpha[\hat{y}_s^n z_s^n - \hat{z}_s^n y_s^n] / ([z_s^n - \hat{z}_s^n] - [y_s^n - \hat{y}_s^n]),$$

11. Replace \hat{z}, \hat{y}, y with the values of \tilde{z}, \hat{z}, z respectively and return to Step 2.

This pseudocode does not include a specific procedure for detecting a stall or cycle, nor for initiating the Global Newton Method in the event of a stall. Its implementation in APL in Appendix A includes subroutines for computing the retraction in Step 2 and the Jacobian in Step 3, and an implementation of the Lemke-Howson algorithm in Step 4. Also included are a program for computing quickly all pure-strategy equilibria, and a recursive program for computing expected payoffs without redundant multiplications.

The table below shows the average time in seconds required by the APL program on a 700 Mhz Windows PC for 900 examples in which both A and g were randomly generated, and an accurate solution was obtained within 1000 seconds. A record of the initial seed for the random number generator enables each example to be reconstructed. In each case the program was initiated with $\hat{z} =$ the vector of 1s so that each $\hat{\sigma}^n$ is the barycenter of Σ_n . The stepsize was $\alpha = 0.02$ for every example. At least 40 examples were solved for each pair (N, m_n) with $N \leq 4$, and at least 20 were solved for each pair with $N > 4$. Each example included in the table was solved with $\|\sigma - \hat{\sigma}\| < 10^{-6}$ and maximum payoff error $< 10^{-6}$.¹

¹A time marked with (* n) indicates a sample truncated by n examples in which convergence with the required accuracy was obtained in 1,000 to 10,000 seconds. In particular, for $(N, m_n) = (3, 12), (4, 10), (7, 4), (12, 2)$ respectively, there were 10, 29, 56, 20 additional examples that converged within 10,000 seconds, and the average time for the full sample was 634, 1046, 2807, 1465 seconds. A total of 30 examples among 1032 did not converge with the required accuracy within 10,000 seconds (of these 15 were for $(N, m_n) = (4, 10)$, and 12 for $(4, 12)$). Median times average 86% of the mean times shown in the table, and 79% for the full sample. Previous test runs indicated that all the smaller examples would be solved successfully with the larger stepsize $\alpha = 0.05$ in about half the time.

N	m_n	2	4	6	8	10	12	14
3		0.516	0.994	2.474	5.440	18.147	45.164	(*10) 281.116
4		1.139	4.138	21.619	168.658	(*29) 405.898		
5		3.822	19.561	(*1) 405.043				
6		7.989	207.604					
7		19.608	(*56) 638.939					
8		39.666						
9		68.413						
10		192.302						
11	(*2)	468.362						
12	(*20)	788.871						

The long times for large examples are due to the many multiplications required in Step 3 to evaluate the Jacobian of the payoff function for a normal-form game. The time required can be much less when the game is specified in extensive form. Also, the times required to reach pure-strategy equilibria are generally much shorter.

Although the theory indicates that the IPA algorithm can stall, we have not found a definitive example. However, this favorable computational experience does not obviate the role of the Global Newton Method, which we still see as the most powerful and reliable. The IPA algorithm finds only a single equilibrium, whereas the Global Newton Method finds *all* equilibria accessible via the designated ray g . The advantages of finding numerous equilibria (in some examples, over 30) compensate for its slower speed, provided a fast start is obtained from the IPA algorithm. Our view, therefore, is that the IPA algorithm is best interpreted as a fast start for the Global Newton Method.

REFERENCES

- [1] Bochnak, J., M. Coste, and M-F. Roy (1998), *Real Algebraic Geometry*. Berlin: Springer-Verlag.
- [2] Eaves, B. (1972), "Homotopies for the Computation of Fixed Points," *Mathematical Programming*, 3, 25-237.
- [3] Eaves, B. (1984), *A Course in Triangulations for Solving Equations with Deformations*. Berlin: Springer-Verlag.
- [4] Eaves, B., and K. Schmedders (1999), "General Equilibrium Models and Homotopy Methods," *Journal of Economic Dynamics and Control*, 23, 1249-1279.
- [5] Eaves, B., and H. Scarf (1981), "Equivalence of LCPs and PLSs," *Mathematics of Operations Research*, 6, 475-494.
- [6] Govindan, S., and R. Wilson (2001), "A Global Newton Method to Compute Nash Equilibria," 1996, revised and resubmitted February 2001.
- [7] Gül, F., D. Pearce, and E. Stachetti (1993), "A Bound on the Proportion of Pure Strategy Equilibria in Generic Games," *Mathematics of Operations Research*, 18, 548-552.
- [8] Howson, J., and R. Rosenthal (1974), "Bayesian Equilibria of Finite Two-Person Games with Incomplete Information," *Management Science*, 21, 313-315.
- [9] Keenan, D. (1981), "Further Remarks on the Global Newton Method," *Journal of Mathematical Economics*, 8, 159-165.
- [10] Kohlberg, E., and J.-F. Mertens (1986), "On The Strategic Stability of Equilibria," *Econometrica*, 54, 1003-39.
- [11] Lemke, C. (1965), "Bimatrix Equilibrium Points and Mathematical Programming," *Management Science*, 11, 681-689.
- [12] Lemke, C., and J. Howson (1964), "Equilibrium Points of Bimatrix Games," *Journal of the Society of Industrial and Applied Mathematics*, 12, 413-423.
- [13] McKelvey, R.D., and A. McLennan (1996), "Computation of Equilibria in Finite Games," in H. M. Amman, D.A. Kendrick, and J. Rust (eds.), *Handbook of Computational Economics*, Volume I, 87-142.
- [14] McKelvey, R., A. McLennan, and T. Turocy (1996), *Gambit*, <http://www.hss.caltech.edu/gambit>.
- [15] McKelvey, R., and T. Palfrey (1995), "Quantal Response Equilibria for Normal Form Games," *Games and Economic Behavior*, 10, 6-38.
- [16] Smale, S. (1976), "A Convergent Process of Price Adjustment and Global Newton Methods," *Journal of Mathematical Economics*, 3, 107-120.
- [17] Rosenmüller, J. (1971), "On a Generalization of the Lemke-Howson Algorithm to Noncooperative N-Person Games," *SIAM Journal of Applied Mathematics*, 21, 73-79.
- [18] Scarf, H., and T. Hansen (1973), *The Computation of Economic Equilibrium*. New Haven: Yale University Press.
- [19] Shapley, L. (1974), "A Note on the Lemke-Howson Algorithm," *Mathematical Programming Study*, 1, 175-189.
- [20] van den Elzen, A., and D. Talman (1991), "A Procedure for Finding Nash Equilibria in Bi-Matrix Games," *ZOR-Methods and Models of Operations Research*, 35, 27-43.
- [21] van den Elzen, A., and D. Talman (1994), "Finding a Nash Equilibrium in Noncooperative N-Person Games by Solving a Sequence of Linear Stationary Point Problems," *ZOR-Mathematical Methods of Operations Research*, 39, 365-375.
- [22] von Stengel, B. (1999), "New Maximal Numbers of Equilibria in Bimatrix Games," *Discrete and Computational Geometry*, 21, 557-568.
- [23] Wilson, R. (1971) "Computing Equilibria of N-Person Games," *SIAM Journal of Applied Mathematics*, 21, 80-87.
- [24] Wilson, R. (1992), "Computing Simply Stable Equilibria," *Econometrica*, 60, 1039-1070.

DEPARTMENT OF ECONOMICS, THE UNIVERSITY OF WESTERN ONTARIO, LONDON, ONTARIO, CANADA N6A 5C2
E-mail address: sgovinda@uwo.ca

STANFORD BUSINESS SCHOOL, STANFORD UNIVERSITY, STANFORD, CALIFORNIA, USA 94305-5015
E-mail address: rwilson@stanford.edu