ZOR – Methods and Models of Operations Research (1991) 35: 27–43

A Procedure for Finding Nash Equilibria in Bi-Matrix Games

By A. H. van den Elzen[1,2] and A. J. J. Talman[2]

Abstract: In this paper we consider the computation of Nash equilibria for noncooperative bi-matrix games. The standard method for finding a Nash equilibrium in such a game is the Lemke-Howson method. That method operates by solving a related linear complementarity problem (LCP). However, the method may fail to reach certain equilibria because it can only start from a limited number of strategy vectors. The method we propose here finds an equilibrium by solving a related stationary point problem (SPP). Contrary to the Lemke-Howson method it can start from almost any strategy vector. Besides, the path of vectors along which the equilibrium is reached has an appealing game-theoretic interpretation. An important feature of the algorithm is that it finds a perfect equilibrium when at the start all actions are played with positive probability. Furthermore, we can in principle find all Nash equilibria by repeated application of the algorithm starting from different strategy vectors.

Zusammenfassung: In diesem Beitrag entwickeln wir ein neues Verfahren zur Berechnung eines Nash-Gleichgewichtspunktes für Zweimatrizen-Spiele. Das Standardverfahren zur Lösung dieser Spiele ist der Lemke-Howson Algorithmus. Dieses Pivotverfahren löst ein lineares Komplementaritätsproblem. Da dieses Verfahren nur in einer beschränkten Anzahl Punkte beginnen kann, können nicht alle Gleichgewichtspunkte berechnet werden.
Unser Verfahren löst ein stationäres Punktproblem definiert auf die Strategiemenge und darf in einem beliebigen Punkt angefangen werden. Der Weg der Punkte des Algorithmus hat eine einladende spieltheoretische Interpretation. Wenn im Startpunkt alle Strategien mit positiver Wahrscheinlichkeit gewählt werden, dann ist das gefundene Nash-Gleichgewicht perfekt. Auch dürfen alle Gleichgewichtspunkte berechnet werden durch mehrere Startpunkte zu wählen.

Key Words: bi-matrix game, Nash equilibrium, complementarity

This research is part of the VF-program "Competition and Cooperation", which has been approved by the Netherlands Ministery of Education and Sciences.

[1] This author is financially supported by the Co-operation Centre Tilburg and Eindhoven Universities, The Netherlands.

[2] A. H. van den Elzen and A. J. J. Talman, Department of Econometrics, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands.

0340-9422/91/1/27–43 $2.50 © 1991 Physica-Verlag, Heidelberg

1 Introduction

In this paper we consider the problem of finding Nash equilibria in mixed strategies for a noncooperative bi-matrix game. A noncooperative bi-matrix game is a two-person game with a finite set of actions for each player. The payoffs of a player are listed in a matrix. The Nash equilibrium is the standard equilibrium concept for a noncooperative game. It states that a strategy is an equilibrium strategy when no player can improve upon his situation by deviating from his strategy while all other players keep on playing their strategies.

The standard method for solving a bi-matrix game is the Lemke-Howson method (see [4]). That method finds a Nash equilibrium by solving a related linear complementarity problem (LCP), which is not defined on the strategy space but on a nonnegative orthant. Each solution then uniquely determines an equilibrium on the strategy space.

The method proposed in this paper finds a Nash equilibrium by solving a stationary point problem (SPP) on the strategy space. Thus, contrary to the Lemke-Howson procedure it directly operates on the strategy space. Both methods are complementary pivoting algorithms and therefore find under some nondegeneracy assumption a positively indexed equilibrium (see [6]). However, because the choice of the starting vector for the Lemke-Howson procedure is restricted, that method may fail to reach certain positively indexed equilibria. In our method the choice of the starting vector is free and therefore we can find in principle all positively indexed equilibria by a repeated application of the algorithm from different starting vectors. We also show how the algorithm can find negatively indexed equilibria by restarting it from the positively indexed equilibria already being found. Our method can be seen as a strategy adjustment process having an appealing game-theoretic interpretation. In this respect it is also interesting that the algorithm finds a perfect equilibrium whenever the starting vector lies in the interior of the strategy space, i.e., when at the start all actions are played with a positive probability.

The organization of the paper is as follows. In Section 2 we show that the set of Nash equilibria in a bi-matrix game can be seen as the solution set of a stationary point problem. Furthermore, we give some intuition concerning the working of the algorithm. The formal steps of the procedure are presented in Section 3. Besides, in that section we give conditions under which the algorithm converges and we prove the perfectness of the equilibrium found by the procedure when starting from a strategy vector in the interior of the strategy space. In Section 4 we present some examples and provide a game-theoretic interpretation of the method. In Section 5 we show how the algorithm can find negatively indexed equilibria. We also give an example of a game for which our method can find more equilibria than the Lemke-Howson method can find. Finally, some remarks are made concerning the computational speed of both algorithms.

2 Solving the Bi-Matrix Game as a Stationary Point Problem

A bi-matrix game is a tuple (n_1, n_2, A, B) in which $n_1(n_2)$ denotes the number of actions of player 1(2) while $A(B)$ is the payoff matrix of player 1(2). The actions of player 1(2) are indexed by $(1, k)$, $k \in \{1, \ldots, n_1\}$ ($(2, k)$, $k \in \{1, \ldots, n_2\}$). The matrices A and B have dimension n_1 x n_2. An element a_{jk} of A denotes the payoff to player 1 if he plays his j-th action while player 2 plays his k-th action. Similarly, b_{jk} denotes the payoff of player 2 in that situation.

The strategy space of player $j, j \in \{1, 2\}$, is defined as the $(n_j - 1)$-dimensional unit simplex $S^{n_j-1} := \{x_j \in \mathbb{R}_+^{n_j} \mid \sum_{k=1}^{n_j} x_{jk} = 1\}$. A vector x_j in S^{n_j-1}, $j \in \{1, 2\}$, represents the mixed strategy of player j at which he plays his k-th action with probability x_{jk}. The n_j vertices of S^{n_j-1} are the unit vectors in $\mathbb{R}^{n_j}$. They correspond to the pure strategies of player j. The strategy space of the game, S, is equal to $S^{n_1-1} \times S^{n_2-1}$. A vector x in S is called a strategy vector and is equal to (x_1, x_2) with $x_1 \in S^{n_1-1}$ and $x_2 \in S^{n_2-1}$. Each vertex of S corresponds to a strategy vector at which both players play a pure strategy. The set S being the product of two unit simplices is called a simplotope.

On the set S we define the function z: $S \to \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ as follows. For each x in S the vector $z(x) = (z_1(x), z_2(x))$ with $z_1(x) \in \mathbb{R}^{n_1}$, $z_2(x) \in \mathbb{R}^{n_2}$, is defined by

$$z_1(x) = Ax_2 \text{ and } z_2(x) = B^{\mathsf{T}} x_1 . \tag{2.1}$$

The element $z_{1k}(x)$, $k \in \{1, \ldots, n_1\}$, is the payoff to player 1 when he plays his k-th action while player 2 plays x_2. Similarly, $z_{2k}(x)$, $k \in \{1, \ldots, n_2\}$, is the payoff to player 2 when he plays his k-th action while player 1 plays x_1. An action (j, k) for which $z_{jk}(x) = \max_h z_{jh}(x)$ is called optimal for player j at x. The number $x_j^{\mathsf{T}} z_j(x)$ is the expected payoff to player $j, j \in \{1, 2\}$, when the strategy vector x is played.

A strategy vector x^* in S is called a Nash equilibrium if

$$x_j^{\mathsf{T}} z_j(x^*) \le x_j^{*\mathsf{T}} z_j(x^*),\ x_j \in S^{n_j-1},\ j \in \{1, 2\} . \tag{2.2}$$

Thus, deviating from x_j^* will not increase the expected payoff to player j. The problem of finding an x^* in S such that x^* satisfies (2.2) is known as the stationary point problem (SPP) on S with respect to z. Since $x_j^{\mathsf{T}} z_j(x^*)$ is linear in x_j we can restrict ourselves in (2.2) to the n_j vertices of S^{n_j-1}, $j \in \{1, 2\}$, so that x^* in S is a Nash equilibrium if and only if $z_{jk}(x^*) \le x_j^{*\mathsf{T}} z_j(x^*)$, $k \in \{1, \ldots, n_j\}$, $j \in \{1, 2\}$. From this it is straightforward to derive that x^* is a Nash equilibrium if and only if for all $k \in \{1, \ldots, n_j\}$,

$$z_{jk}(x^*) = \max_h z_{jh}(x^*) \text{ when } x_{jk}^* > 0,\ j \in \{1, 2\} . \tag{2.3}$$

Thus, at a Nash equilibrium only actions that are optimal for a player are played with positive probability. Note that at a Nash equilibrium x^* the expected payoff to player j is equal to $\max_h z_{jh}(x^*)$ since $z_{jk}(x^*) < \max_h z_{jh}(x^*)$ implies $x^*_{jk} = 0$.

The algorithm searches for a solution to the SPP as given in (2.3). Starting from an arbitrarily chosen strategy vector $v = (v_1, v_2)$ in S, the algorithm generates a piecewise linear path of points in S leading from v to a Nash equilibrium. More precisely, points $x = (x_1, x_2) \in S$ on the path generated by the algorithm satisfy the following conditions. For $k \in \{1, \ldots, n_1\}$,

$$\begin{aligned} x_{1k} &= b(x, v).v_{1k} && \text{if } z_{1k}(x) < \max_h z_{1h}(x) \\ x_{1k} &\geq b(x, v).v_{1k} && \text{if } z_{1k}(x) = \max_h z_{1h}(x), \end{aligned}$$

and for $k \in \{1, \ldots, n_2\}$, (2.4)

$$\begin{aligned} x_{2k} &= b(x, v).v_{2k} && \text{if } z_{2k}(x) < \max_h z_{2h}(x) \\ x_{2k} &\geq b(x, v).v_{2k} && \text{if } z_{2k}(x) = \max_h z_{2h}(x), \end{aligned}$$

$$\text{where } 0 \leq b(x, v) := \min_{(j,h)} \left\{ \frac{x_{jh}}{v_{jh}} \middle| v_{jh} > 0 \right\} \leq 1.$$

Observe that $x = v$ satisfies (2.4) with $b(x, v) = 1$. Also each Nash equilibrium x^* satisfies (2.4) with $b(x^*, v) = 0$ or with $v_{jk} = 0$ for all (j, k) for which $z_{jk}(x^*) < \max_h z_{jh}(x^*)$. In the latter case the non-optimal actions are already played with probability zero at the start. In both cases $x^*_{jk} = b(x^*, v).v_{jk}$ is equal to zero when at x^* action (j, k) is non-optimal. Under some nondegeneracy condition the set of points satisfying (2.4) contains a piecewise linear path, P, from v to a Nash equilibrium. This path will be followed by the algorithm. The notion of nondegeneracy will be made precise further on, but it is for example required that at $x = v$ both $\max_k z_{1k}(x)$ and $\max_\ell z_{2\ell}(x)$ are attained for a unique index. Thus, at the starting strategy vector v each player has only one optimal action. Suppose these maxima are attained for the actions $(1, k_1)$ and $(2, k_2)$, respectively. Clearly, from v, along P, $b(x, v)$ must decrease from 1. Thus, according to (2.4), initially vectors x in S are generated such that all the x_{1k}, $k \neq k_1$, and x_{2k}, $k \neq k_2$, are relatively decreased ($x_{ik} = b(x, v).v_{ik}$, $(i, k) \neq (1, k_1), (2, k_2)$), while both x_{1k_1} and x_{2k_2} are increased in order to keep x_1 in S^{n_1-1} and x_2 in S^{n_2-1}. This is done till $b(x, v)$ becomes 0 and a Nash equilibrium is reached, or till a point x is reached at which $z_{jk}(x) = z_{jk_j}(x)$ for some (j, k), $k \neq k_j$. Then x_{jk} is also relatively increased. In general the algorithm generates strategy vectors x such that all the x_{jk}/v_{jk}, related to the indices (j, k) for which $z_{jk}(x) < \max_h z_{jh}(x)$, are minimal, i.e., equal to $b(x, v)$ ($x_{jk} = 0$ if $v_{jk} = 0$). As soon as one of these components of $z(x)$, say $z_{j\ell}(x)$,

becomes equal to $\max_h z_{jh}(x)$, then $x_{j\ell}/v_{j\ell}$ is increased from $b(x, v)$ ($x_{j\ell}$ is increased from zero if $v_{j\ell} = 0$), while keeping $z_{j\ell}(x)$ maximal for j. On the other hand, if a vector x is generated such that x_{jr}/v_{jr}, with (j, r) such that $z_{jr}(x) = \max_h z_{jh}(x)$, becomes minimal, i.e., equal to $b(x, v)$ (x_{jr} becomes 0 if $v_{jr} = 0$), then vectors y are generated with y_{jr} equal to $b(y, v).v_{jr}$ and $z_{jr}(y)$ is decreased from $\max_h z_{jh}(y)$.

To illustrate the foregoing let us provide a simple example of a game in which both players have two actions and the payoff matrices A and B are given by

$$A = \begin{bmatrix} 4 & 0 \\ -1 & 3 \end{bmatrix} \text{ and } B = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix}.$$

The strategy space for this game is equal to $S = S^1 \times S^1 = \{x \in \mathbb{R}^4_+ \mid x_{11} + x_{12} = 1$ and $x_{21} + x_{22} = 1\}$, and is displayed in Figure 2.1.

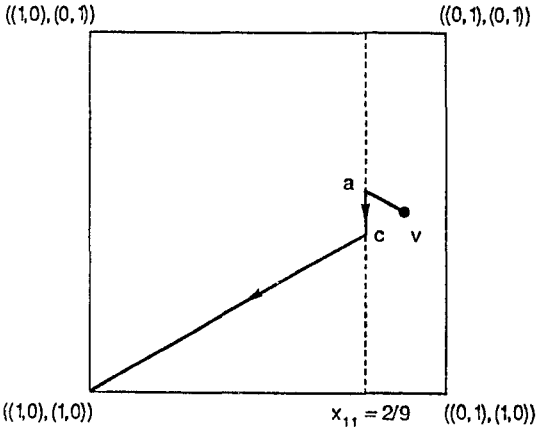

Fig. 2.1. The path P of points satisfying (2.4) from v to a Nash equilibrium.

Consider the starting point $v = ((1/8, 7/8),(1/2, 1/2))$. It is straightforward to verify that $z(v) = ((2, 1), (9/4, 25/8))$. Thus at v, $z_{11}(v)$ and $z_{22}(v)$ are maximal. So, the algorithm leaves v in the direction of $((1, 0),(0, 1))$, i.e., x_{11} and x_{22} are both increased. The algorithm continues in this way till the vector $a = ((2/9, 7/9), (4/9, 5/9))$ is reached at which $z(a) = ((16/9, 11/9), (22/9, 22/9))$, i.e., at $x = a$, $z_{21}(x)$ has become equal to $z_{22}(x)$. Observe that along the segment $[v, a]$ the number $b(x, v)$ decreases from 1 to 8/9. Next the algorithm continues, according to (2.4), by increasing x_{21} relatively away from x_{12} while keeping $z_{21}(x)$ equal to $z_{22}(x)$. It is easy to verify that the latter holds along the line segment $[a, c]$. In fact $z_{21}(x) = z_{22}(x) = 22/9$ if $x_{11} = 2/9$. At $c = ((2/9, 7/9), (5/9, 4/9))$ with $z(c) = ((20/9, 7/9), (22/9, 22/9))$ we still have $z_{21}(c) = z_{22}(c)$ and $z_{11}(c) > z_{12}(c)$. But observe that at $x = c$, x_{22}/v_{22} has become equal to $b(x, v) = x_{12}/v_{12} = 8/9$.

When keeping $z_{22}(x)$ equal to $z_{21}(x)$ one would have to generate points x with $x_{22}/v_{22} < x_{12}/v_{12}$, which contradicts (2.4). In that case one would leave S at the vector ((2/9, 7/9), (1, 0)). But instead the algorithm continues by keeping x_{22} relatively minimal, i.e., x_{22}/v_{22} is kept equal to x_{12}/v_{12}, while, according to (2.4), $z_{22}(x)$ is decreased from $\max_h z_{2h}(x) = z_{21}(x)$. In this way the algorithm reaches the vertex ((1, 0), (1, 0)) which is a pure Nash equilibrium with $z((1, 0),(1, 0)) = ((4, -1),(4, -3))$.

The idea behind the procedure is to generate a sequence of vectors x along which the set T of actions (j, k) for which $z_{jk}(x) = \max_h z_{jh}(x)$ grows while the probabilities related to all the other actions are driven down to zero, since if they are zero, a Nash equilibrium is found, as follows from (2.3). However, the set T does not need to grow monotonically (cf. point c in the foregoing example). This guarantees the convergence of the algorithm to a Nash equilibrium.

3 The Procedure

The algorithm is a complementary pivoting procedure. To implement it we first have to rewrite (2.4) into a system of linear equations. By substituting (2.1) we obtain that the process generates from v the (piecewise linear) path P of strategy vectors $x \in S$ satisfying for $k \in \{1, \ldots, n_1\}$,

$$\begin{aligned} x_{1k} &= bv_{1k} && \text{if } A^k x_2 < \beta_1 \\ x_{1k} &\geq bv_{1k} && \text{if } A^k x_2 = \beta_1 , \end{aligned}$$

and for $k \in \{1, \ldots, n_2\}$, (3.1)

$$\begin{aligned} x_{2k} &= bv_{2k} && \text{if } x_1^{\mathrm{T}} B_k < \beta_2 \\ x_{2k} &\geq bv_{2k} && \text{if } x_1^{\mathrm{T}} B_k = \beta_2 , \end{aligned}$$

where $b := b(x, v)$, C^k and C_k respectively are the k-th row and k-th column of a matrix C, $\beta_1 = \max_k A^k x_2$, and $\beta_2 = \max_k x_1^{\mathrm{T}} B_k$.

Before going further we introduce some additional notation. The set of actions of player j is denoted by $I(j), j \in \{1, 2\}$. Thus, $I(j) = \{(j, 1), \ldots, (j, n_j)\}$. The set $I = I(1) \cup I(2)$ denotes the set of all actions in the game. For each x in S satisfying (3.1) there is at least one set $T \subset I$ such that $x_{jk} \geq bv_{jk}$ and $z_{jk}(x) = \beta_j$ for all $(j, k) \in T$ while $z_{ih}(x) \leq \beta_i$ and $x_{ih} = bv_{ih}$ for all $(i, h) \notin T$. From this observation we obtain that the procedure generates for a sequence of subsets T of I with $T_j := T \cap I(j) \neq \emptyset$, starting from $x = v$, strategy vectors x in S such that

$$\begin{aligned} x_{1k} &= bv_{1k} && \text{and } A^k x_2 \leq \beta_1 && \text{if } (1, k) \notin T \\ x_{1k} &\geq bv_{1k} && \text{and } A^k x_2 = \beta_1 && \text{if } (1, k) \in T , \end{aligned}$$

and (3.2)

$$\begin{aligned} x_{2k} &= bv_{2k} && \text{and } x_1^{\mathsf{T}} B_k \le \beta_2 && \text{if } (2,k) \notin T \\ x_{2k} &\ge bv_{2k} && \text{and } x_1^{\mathsf{T}} B_k = \beta_2 && \text{if } (2,k) \in T. \end{aligned}$$

We now show how the path P can be followed by a sequence of linear programming (ℓ.p) pivot steps in a system of linear equations obtained from (3.2). Denote the set of points x in S satisfying (3.2) for certain $T \subset I$ by $B(T)$. Thus, the algorithm reaches a Nash equilibrium via a (finite) sequence of sets $B(T), T \subset I$. Assuming nondegeneracy as given below, each nonempty $B(T)$ is a line segment in S and we need one ℓ.p step to traverse such a $B(T)$. The system of linear equations in which the ℓ.p pivot step is made is obtained from (3.2) by introducing slack variables for each inequality. The slack variables for the inequalities $A^h x_2 \le \beta_1, (1,h) \notin T$, are denoted by μ_{1h} and those for $x_1^{\mathsf{T}} B_h \le \beta_2, (2,h) \notin T$, by μ_{2h}. The slack variables for the inequalities $x_{jk} \ge bv_{jk}$, $(j,k) \subset T$, are denoted by λ_{jk}. Adding these slacks to (3.2) we obtain that x belongs to $B(T)$ iff for $j \in \{1,2\}$ there exist $\lambda_{jk} \ge 0$ for $(j,k) \in T$, $\mu_{jk} \ge 0$ for $(j,k) \notin T$, $\beta_j \in \mathbb{R}$ and $0 \le b \le 1$, such that

$$x_j = bv_j + \sum_{(j,k)\in T_j} \lambda_{jk} e_j(k) \quad \text{with} \quad \sum_{(j,k)\in T_j} \lambda_{jk} = 1 - b, \tag{3.3}$$

while

$$Ax_2 + \sum_{(1,h)\notin T} \mu_{1h} e_1(h) = e_1 \beta_1$$

and (3.4)

$$B^{\mathsf{T}} x_1 + \sum_{(2,h)\notin T} \mu_{2h} e_2(h) = e_2 \beta_2 .$$

Here $e_j(k)$ is the k-th unit vector in $\mathbb{R}^{n_j}$ while e_j is the vector of ones in $\mathbb{R}^{n_j}$, $j \in \{1,2\}$.

Substituting (3.3) in (3.4) gives the system of linear equations

$$b\begin{pmatrix} Av_2 \\ B^{\mathsf{T}} v_1 \\ 1 \\ 1 \end{pmatrix} + \sum_{(1,k)\in T} \lambda_{1k} \begin{pmatrix} \underline{0} \\ B_k^{\mathsf{T}} \\ 1 \\ 0 \end{pmatrix} + \sum_{(2,k)\in T} \lambda_{2k} \begin{pmatrix} A_k \\ \underline{0} \\ 0 \\ 1 \end{pmatrix} + \sum_{(1,h)\notin T} \mu_{1h} \begin{pmatrix} e_1(h) \\ \underline{0} \\ 0 \\ 0 \end{pmatrix}$$

$$+\sum_{(2,h)\notin T}\mu_{2h}\begin{pmatrix}\underline{0}\\ e_2(h)\\ 0\\ 0\end{pmatrix}-\beta_1\begin{pmatrix}e_1\\ \underline{0}\\ 0\\ 0\end{pmatrix}-\beta_2\begin{pmatrix}\underline{0}\\ e_2\\ 0\\ 0\end{pmatrix}=\begin{pmatrix}\underline{0}\\ \underline{0}\\ 1\\ 1\end{pmatrix}, \tag{3.5}$$

where $\underline{0}$ denotes a vector of zeros of appropriate length. In the system (3.5) there are n_1+n_2+2 equations and n_1+n_2+3 variables. The last two equations of (3.5) reflect the property that $\sum_{(j,k)\in T_j}\lambda_{jk}=1-b$, $j\in\{1,2\}$. These equations can be eliminated by substituting for $j\in\{1,2\}$ one of the λ_{jk}'s, say λ_{jk_j}, by

$$\lambda_{jk_j}=1-b-\sum_{\substack{(j,k)\in T_j\\ k\neq k_j}}\lambda_{jk}\,. \tag{3.6}$$

Let T_j^1, $j\in\{1,2\}$, be the set of indices defined by $T_j^1=T_j\setminus\{(j,k_j)\}$ and let $T^1=T_1^1\cup T_2^1$. All of this together and the substitution of b by $1-b'$, gives the following system of equations

$$\begin{aligned}&b'\begin{pmatrix}A_{k_2}-Av_2\\ B_{k_1}^{\mathcal{T}}-B^{\mathcal{T}}v_1\end{pmatrix}+\sum_{(1,k)\in T^1}\lambda_{1k}\begin{pmatrix}\underline{0}\\ B_k^{\mathcal{T}}-B_{k_1}^{\mathcal{T}}\end{pmatrix}+\sum_{(2,k)\in T^1}\lambda_{2k}\begin{pmatrix}A_k-A_{k_2}\\ \underline{0}\end{pmatrix}\\ &+\sum_{(1,h)\notin T}\mu_{1h}\begin{pmatrix}e_1(h)\\ \underline{0}\end{pmatrix}+\sum_{(2,h)\notin T}\mu_{2h}\begin{pmatrix}\underline{0}\\ e_2(h)\end{pmatrix}-\beta_1\begin{pmatrix}e_1\\ \underline{0}\end{pmatrix}\\ &-\beta_2\begin{pmatrix}\underline{0}\\ e_2\end{pmatrix}=\begin{pmatrix}-Av_2\\ -B^{\mathcal{T}}v_1\end{pmatrix}.\end{aligned} \tag{3.7}$$

The latter system has only n_1+n_2 equations and n_1+n_2+1 variables, and the variables must satisfy $\lambda_{jk}\geq 0$ for $(j,k)\in T^1$, $\sum_{(j,k)\in T_j^1}\lambda_{jk}\leq b'$ for $j\in\{1,2\}$, $0\leq b'\leq 1$, $\mu_{ih}\geq 0$ for $(i,h)\notin T$. A solution to this system is denoted by $(b',\lambda_1,\lambda_2,\mu_1,\mu_2,\beta_1,\beta_2)$ and corresponds to a point $x=(x_1,x_2)$ in $B(T)$ as defined in (3.3). The $\ell.p$ pivot steps of the algorithm are made in system (3.7).

Before stating the formal steps of the algorithm we give a condition that guarantees the convergence of the procedure. This condition is explained in more detail after the description of the algorithm.

Assumption 3.1.: (nondegeneracy assumption). At each solution $(b',\lambda_1,\lambda_2,\mu_1,\mu_2,\beta_1,\beta_2)$ of (3.7) at most one of the constraints $0\leq b'\leq 1$, $\lambda_{jk}\geq 0$ for $(j,k)\in T^1$, $b'\geq\sum_{(j,k)\in T_j^1}\lambda_{jk}$, $\mu_{ih}\geq 0$ for $(i,h)\notin T$, is binding unless $b'=1$ or $v_{ih}=0$ for all $(i,h)\notin T$.

Step 0: Initialization
Choose an arbitrary vector v in S. If v is a Nash equilibrium then the algorithm stops. Else calculate the (unique) indices $(1, k_1)$ and $(2, k_2)$ for which $A^{k_1}v_2 = \max_k A^k v_2$ and $v_1^T B_{k_2} = \max_k v_1^T B_k$. Furthermore, set $T^1 = 0, b' = 0\beta_1 = A^{k_1}v_2, \beta_2 = v_1^T B_{k_2}, \mu_{1h} = \beta_1 - A^h v_2$ for $h \neq k_1$ and $\mu_{2h} = \beta_2 - v_1^T B_h$ for $h \neq k_2$. Increase b' from 0 in (3.7) and go to Step 1.

Step 1:

a. If b' becomes 1 then let the solution of (3.7) be $(1, \lambda_1^*, \lambda_2^*, \mu_1^*, \mu_2^*, \beta_1^*, \beta_2^*)$. The vector $x^* = (x_1^*, x_2^*)$, with $x_j^* = \sum_{(j,k)\in T_j} \lambda_{jk}^* e_j(k)$, $j \in \{1, 2\}$, is a Nash equilibrium and the algorithm stops.
b. If λ_{jk} becomes 0 for some $(j, k) \in T^1$ then T^1 becomes $T^1 \setminus \{(j, k)\}$ and go to Step 2a.
c. If $\sum_{(j,k)\in T_j^1} \lambda_{jk}$ becomes equal to b' for some $j \in \{1,2\}$ then according to (3.6), λ_{jk_j} becomes 0. Go to Step 2b.
d. If μ_{ih} becomes zero for some $(i, h) \notin T$ then go to Step 3.

Step 2:

a. Increase the complementary variable μ_{jk} from zero by pivoting into system (3.7) the column $(e_1(k), \underline{0})^T$ if $j = 1$, or $(\underline{0}, e_2(k))^T$ if $j = 2$. Return to Step 1.
b. Substitute the largest λ_{jk}, say $\lambda_{j\ell}$, by $b' - \sum_{\substack{(j,h)\in T_j \\ h\neq k_j, \ell}} \lambda_{jh}$. Increase μ_{jk_j} from zero by pivoting the related column into system (3.7). T^1 becomes $T^1 \setminus \{(j, \ell)\}$, k_j becomes ℓ, and return to Step 1.

Step 3:
If additionally $v_{jk} = 0$ for all $(j, k) \notin T \cup \{(i, h)\}$ then let the solution be $(b', \lambda_1^*, \lambda_2^*, \mu_1^*, \mu_2^*, \beta_1^*, \beta_2^*)$. The vector $x^* = (x_1^*, x_2^*)$, with $x_j^* = (1 - b')v_j + \sum_{(j,k)\in T_j} \lambda_{jk}^* e_j(k)$, $j \in \{1, 2\}$, is a Nash equilibrium and the algorithm stops.
Else increase the complementary variable λ_{ih} from zero by pivoting its related column into system (3.7). T^1 becomes $T^1 \cup \{(i, h)\}$ and return to Step 1.

Let us make a few remarks. The algorithm starts with increasing b' from zero. From (3.6) we derive that this means that both λ_{1k_1} and λ_{2k_2} are increased from zero. In Step 1c the variable λ_{jk_j} becomes zero. Then we have to adapt system (3.7) by eliminating another λ_{jk} to take over the role of λ_{jk_j}. In principle any λ_{jk}, $(j, k) \in T_j^1$, can be taken. We suggest to take the largest, say $\lambda_{j\ell}$. This substitution can easily be performed in (3.7) by adding the column related to $\lambda_{j\ell}$ to the column related to b' and subtracting the same column from the columns related to λ_{jh}, $h \neq \ell$.

Assumption 3.1 is standard in linear programming and assures that all steps of the algorithm are unique. More precisely, when this nondegeneracy condition holds it

cannot occur that more than one constraint in Step 1 becomes binding simultaneously. This is only allowed when the algorithm stops. Assumption 3.1 also guarantees the uniqueness of the indices $(1, k_1)$ and $(2, k_2)$ in Step 0. To see this, suppose that $A^{k_1}v_2 = \max_k A^k v_2 = A^\ell v_2$ for some $\ell \neq k_1$. But then $\mu_{1\ell} = 0$ and two constraints (the other one is $b' = 0$) are binding. This is the only restriction on the choice of the starting vector.

The nondegeneracy condition guarantees the convergence of the algorithm. First, observe that the solution set of (3.7) is bounded for given T. Since $\beta_j = \max_k z_{jk}(x)$ and z is a linear function on the compact set S, β_j must be finite and therefore also the μ_{ih}'s with $(i, h) \notin T$. This together with Assumption 3.1 implies that for each T the solution set of (3.7) and hence also $B(T)$ is either empty or a line segment with two end points. The algorithm starts by traversing $B(T^0)$, with $T^0 = \{(1, k_1),(2, k_2)\}$. If the algorithm operates in some $B(T)$ and λ_{jk} becomes zero for some $(j, k) \in T$ then the algorithm continues in $B(T\backslash\{(j, k)\})$. Similarly, the algorithm continues in $B(T \cup \{(i, h)\})$ if μ_{ih} for some $(i, h) \notin T$, becomes zero and no Nash equilibrium is reached. Assumption 3.1 guarantees that these transitions are unique so that no cycling can occur. Because there are only a finite number of possible subsets T the algorithm must reach within a finite number of ℓ.p steps a Nash equilibrium.

What about the relation between our nondegeneracy condition and the nondegeneracy condition on games of Lemke-Howson [4] (See also Shapley [6])? Our condition is much weaker. We only need this condition to hold along the path generated by the algorithm. The last pivot step may be degenerated. In other words, it is possible to apply our procedure to games that are degenerated in the sense of Lemke-Howson. In the next section we give an example.

At the end of this section we discuss whether something can be said about game-theoretic properties holding for the equilibrium found by our algorithm. More concrete, we would like to know whether the equilibrium found is isolated, quasi-strong, regular, essential, perfect, or proper. The precise definition of these concepts can be found for example in van Damme [2]. It turns out that our algorithm finds a perfect equilibrium whenever it starts from an interior strategy vector. This is interesting the more because for bi-matrix games an equilibrium is perfect iff it is undominated (see Theorem 3.2.2 in [2]). Other properties may not hold for the equilibrium reached by our procedure. We remark that this is partly due to the fact that the algorithm can for example be applied to certain games with no isolated equilibria.

Thus, the positive result is that our algorithm finds a perfect equilibrium whenever the starting point v is in the inner of the strategy space, i.e., when $v_{jk} > 0$ for all $(j, k) \in I$. To prove the statement it is most convenient to define a perfect equilibrium as a limit point of a sequence of ε-perfect equilibria. Thus, first we have to define ε-perfectness.

Definition 3.2. The strategy vector $x(\varepsilon) \in S$ is an ε-perfect equilibrium if it is completely mixed and satisfies

$$\text{if } z_{jk}(x(\varepsilon)) < z_{j\ell}(x(\varepsilon)) \text{ then } x_{jk}(\varepsilon) \leq \varepsilon, \forall j, k, \ell.$$

Observe that an ε-perfect equilibrium need not to be a Nash-equilibrium. The concept only states that non-optimal actions are played with a small probability, i.e., players make only small mistakes. Now x^* in S is a *perfect equilibrium* if x^* is a limit point of a sequence $\{x(\varepsilon)\}_{\varepsilon \downarrow 0}$, where for all ε, $x(\varepsilon)$ is an ε-perfect equilibrium.

Theorem 3.3.: If v lies in the interior of S then the algorithm finds a perfect Nash equilibrium.

Proof: Recall that the vectors x on the path generated by the algorithm satisfy

$$x_{jk} = b(x,v).v_{jk} \qquad \text{if } z_{jk}(x) < \max_{\ell} z_{j\ell}(x)$$
$$x_{jk} \geq b(x,v).v_{jk} \qquad \text{if } z_{jk}(x) = \max_{\ell} z_{j\ell}(x),$$

$$\text{where } 0 \leq b(x, v) := \min_{(i,h)} \left\{ \frac{x_{ih}}{x_{ih}} \mid v_{ih} > 0 \right\} \leq 1.$$

First of all, every Nash equilibrium x^* in the inner of S is perfect for then $z_{jk}(x^*) = \max_{\ell} z_{j\ell}(x^*)$ for all $(j, k) \in I$. Next, because $v_{jk} > 0$ for all $(j, k) \in I$, the algorithm can only reach an equilibrium x^* on the boundary of S if in the last iteration of the algorithm $b(x, v)$ becomes zero. Suppose $b(x, v)$ decreases along the line segment $[y, x^*]$ from $b(y, v)$ down to zero. Let $I(x^*) = \{(j, k) \in I \mid x^*_{jk} = 0\}$ and let (r, ℓ) be an index for which $y_{r\ell} = \max_{(i,p) \in I(x^*)} y_{ip}$. Now each vector x on (y, x^*) is an ε-perfect equlibrium $x(\varepsilon)$ with $\varepsilon = x_{r\ell}$. The limit point x^* of $\{x(\varepsilon)\}_{\varepsilon \downarrow 0}$ is perfect by definition. □

Observe that we not only proved that our algorithm finds a perfect equilibrium but also that the last linear piece of the path consists of ε-perfect equilibria. Furthermore, we emphasize that the result above is not trivial. This because the algorithm can be applied to games which are degenerated in the sense of Lemke-Howson. For such games not all equilibria are perfect. However, our procedure succeeds to find one when being started in the interior of the strategy space. In Section 4 we illustrate this with an example.

4 Game-Theoretic Interpretation

In this section we want to explain the working of the algorithm in terms of strategies and payoffs. Technically speaking the algorithm roughly works as follows. From the start b' is increased from zero. Recall from (3.6) that an increase of b' means that both λ_{1k_1} and λ_{2k_2} are increased and that b is decreased (from 1). As soon as μ_{jk} for some $(j, k) \notin T$ becomes zero, its complementary variable λ_{jk} is increased from zero and vice versa. From (3.3) we deduce that a positive λ_{jk} means that the relative probability with which player j uses his k-th action is larger than the relative minimum ($x_{jk} > bv_{jk}$), whereas $\lambda_{jk} = 0$ indicates that $x_{jk} = bv_{jk}$. Similarly, from (3.4) we infer that $\mu_{jk} > 0$ corresponds to action (j, k) being non-optimal for player j, i.e., $z_{jk}(x) < \max_h z_{jh}(x)$, whereas $\mu_{jk} = 0$ means that action (j, k) is optimal for player j. With all of this together it is straightforward to derive the game-theoretic interpretation of the adjustments made by the algorithm.

From the start the probabilities related to the unique optimal actions of both players are increased, whereas the probabilities related to all other actions are proportionally decreased. If the latter probabilities all become zero then a Nash equilibrium is reached. This because then all non-optimal actions are played with zero probability. Else, the algorithm eventually generates a strategy vector at which for some player a second action becomes optimal. Then the procedure continues by keeping that action optimal whereas the corresponding probability is relatively increased. In general, the algorithm generates strategy vectors at which the non-optimal actions are played with probabilities all being, relative to the starting probabilities, equal to each other and smaller than each probability with which an optimal action is played. As soon as a non-optimal action becomes optimal, its relative probability is increased from the probabilities related to the non-optimal actions. On the other hand, if a probability with which an optimal action is played becomes relatively equal to the probabilities of the non-optimal actions then it is kept equal to these and the algorithm continues by making the related action non-optimal.

For a more specific illustration we apply the algorithm to the example presented earlier (see Figure 2.1). At v for the solution $(b', \mu_1, \mu_2, \beta_1, \beta_2)$ of (3.7) holds that $\beta_1 = (Av_2)_1 = 2$, $\beta_2 = (B^{\mathrm{T}}v_1)_2 = 25/8$, $\mu_{12} = \beta_1 - (Av_2)_2 = 2 - 1 = 1$, $\mu_{21} = \beta_2 - (B^{\mathrm{T}}v_1)_1 = 25/8 - 9/4 = 7/8$ while $b' = 0$. The algorithm leaves v by increasing b' from zero, i.e., λ_{11} and λ_{22} are increased (Step 0). At the vector a, the variable μ_{21} has become zero (Step 1d) whereas $b' = 1/9$, $\beta_1 = (Aa_2)_1 = 16/9$, $\beta_2 = (B^{\mathrm{T}}a_1)_2 = 22/9$, $\mu_{12} = 5/9$. Thus, the algorithm continues from a by increasing λ_{21} from zero (Step 3). The solution at c is $\beta_1 = (Ac_2)_1 = 20/9$, $\beta_2 = (B^{\mathrm{T}}c_1)_1 = (B^{\mathrm{T}}c_1)_2 = 22/9$, $\mu_{12} = \beta_1 - (Ac_2)_2 = 13/9$, $\lambda_{21} = b' = 1/9$. Thus λ_{22} has become zero (Step 1c). In system (3.7), $(2, k_2)$ becomes $(2, 1)$ and the algorithm continues by increasing μ_{22} from zero (Step 2b). In the next step the algorithm reaches the Nash equilibrium $((1, 0),(1, 0))$ at which $b' = 1$ (Step 1a).

The game-theoretic interpretation of the adjustments along the latter path is as follows. At v, action $(1, 1)$ is optimal for player 1 and action $(2, 2)$ is optimal for player

2. Now the algorithm increases from v the probabilities with which these actions are played and decreases the probabilities of all other actions with the same rate. The algorithm continues in this way till it generates the strategy vector a at which player 2 becomes in equilibrium ($z_{21}(a) = z_{22}(a)$). From a the algorithm generates strategy vectors x by relatively increasing probability x_{21} away from x_{12} while keeping player 2 in equilibrium. At c the probability with which player 2 plays his second action has become relatively equal to the probability related to the only non-optimal action (1, 2). Then the algorithm distorts the equilibrium situation of player 2. It continues by generating vectors x at which for player 2 the action (2, 2) is non-optimal, i.e., $z_{22}(x)$ is made smaller than $z_{21}(x)$. Meanwhile, the probabilities x_{12} and x_{22} are kept relatively equal to each other but smaller than x_{11} and x_{21}. In this way the Nash equilibrium ((1, 0),(1, 0)) is reached.

We conclude this section with an application of our algorithm to a bi-matrix game which is degenerated in the sense of Lemke-Howson [4] (see also [6]). The algorithm cannot only be applied to this game but it also finds a perfect Nash equilibrium whenever it starts from the interior, although there are an infinite number of equilibria not being perfect. The game we consider is the bi-matrix game with payoff matrices

$$A = \begin{bmatrix} 4 & 1 \\ 0 & 1 \end{bmatrix} \qquad \text{and } B = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

This game is graphically represented in Figure 4.1. The piecewise linear curve heavily drawn denotes the set of optimal strategies of player 1 against player 2, also

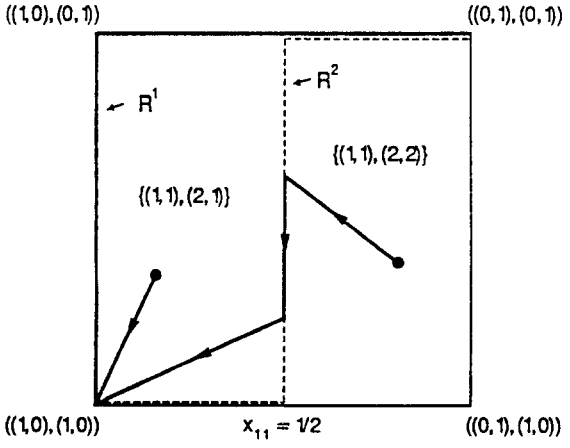

Fig. 4.1. The set of Nash equilibria consists of the unique perfect Nash equilibrium ((1, 0),(1, 0)) and the set $\{x \in S \mid x_{11} \leq 1/2,\ x_2 = (0,1)\}$.

called the best reply set R^1 of player 1. We see that player 1 plays action (1, 1) with probability one against all strategies of player 2, except when player 2 plays (2, 2) with probability one. In that case player 1 is indifferent between his actions. Similarly, one can derive the best reply set R^2 of player 2, indicated in the figure by the dashed piecewise linear curve. The Nash equilibria coincide with the intersections of R^1 and R^2. In the figure $\{(1, 1),(2, 1)\}$ indicates that in the corresponding subset of S the actions (1, 1) and (2, 1) are optimal. Similarly, for $\{(1, 1), (2, 2)\}$.

When the algorithm starts in the interior of S it always finds the perfect equilibrium $((1, 0), (1, 0))$. We have given two possible paths in the figure. Observe that all Nash equilibria except ((1, 0), (1, 0)) fail to be perfect. Consider for example the equilibrium $((0, 1), (0, 1))$. If player 2 makes a mistake and plays his first action with arbitrary small probability then player 1 immediately plays (1, 1). Hence, this equilibrium is unstable against small mistakes.

5 How to Find More Equilibria

Both our algorithm and the Lemke-Howson procedure are complementary pivoting algorithms and therefore find positively indexed equilibria. The notion of the index of an equilibrium in a bi-matrix game has been introduced by Shapley [6]. The index of an equilibrium is positive or negative depending on the sign of the determinant of a matrix related to the payoff structure at that equilibrium. For more details we refer the reader to [6]. In principle our algorithm can find all positively indexed equilibria. This because our procedure can start from almost all strategy vectors. Because the Lemke-Howson method can only start from a limited number of strategy vectors, it is possible that this procedure cannot detect some of the positively indexed equilibria. We illustrate this with an example given by Shapley in the article mentioned. Furthermore, we show that our algorithm can also find negatively indexed equilibria by restarting in a positively indexed equilibrium already found. This can only be done for games with an odd number of isolated equilibria, so that the number of negatively indexed equilibria is one less than the number of positively indexed equilibria (see [6]). This is not a great restriction because the set of bi-matrix games having these properties is dense in the set of all bi-matrix games (see [2]). We conclude this section with some remarks on the computational performance of the Lemke-Howson algorithm and our procedure.

Before giving the example of Shapley we first give a rough impression of the Lemke-Howson procedure. It solves a bi-matrix game (n_1, n_2, A, B) by solving a related linear complementarity problem on $\mathbb{R}_+^{n_1+n_2}$. Any vector $y = (y_1, y_2)$ in $\mathbb{R}_+^{n_1+n_2}$ corresponds to a strategy vector $x = (x_1, x_2)$ in S, where $x_j = y_j(\sum_{\ell=1}^{n_j} y_{j\ell})^{-1}$, $j \in \{1, 2\}$. By this transformation we can indicate how the Lemke-Howson procedure operates on the strategy space S. It starts from a pure strategy vector at which player 1 plays some pure strategy while player 2 plays his best reply against that strategy. If this

vector is not a Nash equilibrium, the algorithm starts with increasing the probability related to the best reply strategy of player 1 against the starting strategy of player 2. In this way, the Lemke-Howson method follows the best reply set of player 2 till it reaches a Nash equilibrium. For more details we refer to [3]. Note that the starting vector is related to one of the pure strategies of player 1. By interchanging the players the maximal number of different starting vectors therefore equals $n_1 + n_2$.

The example of Shapley concerns a 3 x 3 bi-matrix game with payoff matrices

$$A = \begin{bmatrix} 0 & 3 & 0 \\ 2 & 2 & 0 \\ 3 & 0 & 1 \end{bmatrix} \qquad \text{and } B = \begin{bmatrix} 0 & 2 & 3 \\ 3 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Observe that the payoff structure for both players is identical. This game possesses two positively indexed Nash equilibria, i.c. ((0, 0, 1), (0, 0, 1)) and ((1/3, 2/3, 0), (1/3, 2/3, 0)). However, the Lemke-Howson algorithm can only find the first one. This because that method can only start in one of the three vectors ((1, 0, 0), (0, 0, 1)), ((0, 1, 0),(1, 0, 0)) and ((0, 0, 1), (0, 0, 1)), each corresponding to a pure strategy of player 1 and the best reply of 2. Of course, interchanging the players gives no additional starting vectors because the payoffs are equal for both players.

With our algorithm we can also find the second positively indexed Nash equilibrium. For example, when it starts from v = ((0, 1, 0), (0, 1, 0)), it reaches ((1/3, 2/3, 0), (1/3, 2/3, 0)) in one step. This because $z(v) = ((3, 2, 0), (3, 2, 0))$, making that the algorithm generates from v a path of vectors x with $x_{j1} = 1 - b$, $x_{j2} = b$, and $x_{j3} = 0, j \in \{1, 2\}$, with b decreasing from 1 till 2/3.

Next, we indicate how to find negatively indexed equilibria with our algorithm. We recall from Section 2 that the algorithm generates from a starting vector v a piecewise linear path of vectors x in S all lying in the set $B_v := \cup_T B_v(T), T \subsetneq I$, where

$$B_v(T) = \{x \in S \mid x_{jk} \geq b(x, v).v_{jk} \text{ and } z_{jk}(x) = \max_h z_{jh}(x), (j, k) \in T$$
$$x_{jk} = b(x, v).v_{jk} \text{ and } z_{jk}(x) \leq \max_h z_{jh}(x), (j, k) \notin T\}.$$

In general the set B_v is a 1-dimensional manifold. More precisely, for games with an odd number of isolated equilibria it consists of disjoint piecewise linear paths. One such path connects v and a positively indexed Nash equilibrium while all other paths connect two Nash equilibria, one positively and one negatively indexed. In this way all other equilibria are connected.

How can we use the set B_v to find negatively indexed Nash equilibria? Suppose we have found k different positively indexed Nash equilibria by starting our algorithm from k different starting vectors. Then we consider the set B_v related to the starting vector v from which the first Nash equilibrium was found. We successively restart our

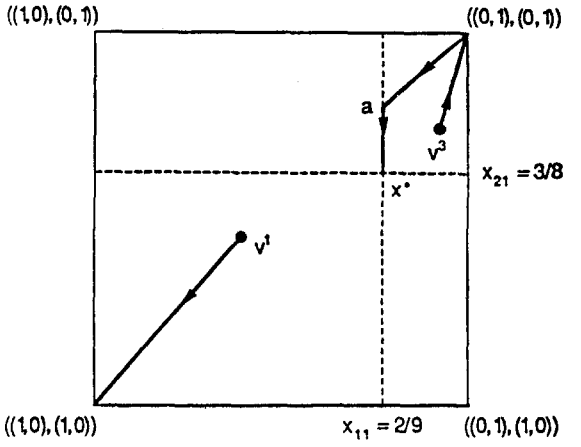

Fig. 5.1. The set B_{v^1} consists of a linear path connecting v^1 and $((1,0),(1,0))$ and a piecewise linear path connecting $((0, 1), 0, 1))$ and x^*. The line segment connecting $((0, 1), (0, 1))$ and a lies on the line through v^1 and $((0, 1), (0, 1))$ and equals $B_{v^1}(\{(1, 2), (2, 2)\})$. Furthermore, the line segment $[a, x^*]$ equals $B_{v^1}(\{(1, 2), (2, 1), (2, 2)\})$.

algorithm from the other $k-1$ positively indexed equilibria and find $k-1$ different negatively indexed equilibria by following the piecewise linear paths in B_v connecting the equilibria with positive and negative index. That these paths can indeed be generated by our algorithm follows from the fact that vectors on the paths satisfy the conditions (2.4). The only additional step is the initialization of the algorithm at a positively indexed equilibrium x^*. In practice this can be done by substituting v for the starting vector v^* related to x^* in the final system of the algorithm starting from v^*.

Again we illustrate this with the example of Section 2. Consider Figure 5.1. When we apply our algorithm starting from v^1 we find $((1, 0), (1, 0))$. Next we apply the algorithm from v^3 and find $((0, 1), (0, 1))$. The third (negatively indexed) equilibrium is found by restarting the algorithm in $((0, 1), (0, 1))$ while considering v^1 as starting vector in the system. Thus, in the final system related to $((0, 1), (0, 1))$ with $v = v^3$ we have to substitute v^1 for v^3. Applying the algorithm in this manner means that we reach $((2/9, 7/9),(3/8, 5/8))$ by traversing in two steps the piecewise linear path in B_{v^1} connecting $((0, 1), (0, 1))$ and $((2/9, 7/9), (3/8, 5/8))$.

To compare the computational strength of our algorithm and the Lemke-Howson procedure we applied both algorithms to several games of different dimension. More precisely, the number of actions of both players varied from 4 to 16. The computational speed was measured in the number of pivot steps needed to obtain an equilibrium. Overall seen, the Lemke-Howson method performs best. In particular that method is in favour for games of small dimension and games in which the number of actions

of one player is small compared to the other. We then let the Lemke-Howson method operate on the best reply set of the player having the largest number of actions. Our method is more suited for relatively large games with both players having a more or less equal number of actions. The reason is that our algorithm adapts all probabilities simultaneously. Of course, the choice of the starting strategy vector is crucial for our algorithm. In general it seems impossible to select a 'best' starting vector from the data. We did experiments with starting vectors related to rationalizable strategies (see [1] and [5]). At such starting vectors only actions are played which are a best reply against an action played by the other player. However, the results are not better than those obtained for starting vectors at which all actions are played with equal probability.

References

[1] Bernhein D (1984) "Rationalizable strategic behaviour", Econometrica 52: 1007–1028
[2] van Damme EEC (1983) Refinements of the Nash Equilibrium Concept, Lecture Notes in Economics and Mathematical Systems 219, Springer Verlag Berlin
[3] van den Elzen A (1990) "Interpretation and generalization of the Lemke-Howson algorithm", Methods of Operations Research 60: 337–345
[4] Lemke CE, Howson JT (1964) "Equilibrium points of bimatrix games", SIAM Journal of Applied Mathematics12: 413–423
[5] Pearce D (1984) "Rationalizable strategic behaviour and the problem of perfection", Econometrica 52: 1029–1040
[6] Shapley LS (1974) "A note on the Lemke-Howson algorithm", Mathematical Programming Studies 1: 175–189

Received October 1988
Revised version received October 1989