Computing normal form perfect equilibria for extensive two-person games

von Stengel, B.; van den Elzen, A.H.; Talman, A.J.J.

Published in:
Econometrica

Publication date:
2002

Document Version
Peer reviewed version

COMPUTING NORMAL FORM PERFECT EQUILIBRIA FOR EXTENSIVE TWO-PERSON GAMES

By Bernhard von Stengel, Antoon van den Elzen, and Dolf Talman[1]

July 31, 2000

This paper presents an algorithm for computing an equilibrium of an extensive two-person game with perfect recall. The method is computationally efficient by using the sequence form, whose size is proportional to the size of the game tree. The equilibrium is traced on a piecewise linear path in the sequence form strategy space from an arbitrary starting vector. If the starting vector represents a pair of completely mixed strategies, then the equilibrium is normal form perfect. Computational experiments compare the sequence form and the reduced normal form, and show that only the sequence form is tractable for larger games.

Keywords: Extensive game, linear complementarity, Nash equilibrium, normal form perfect equilibrium, sequence form.

---

[1]The authors thank Eric van Damme, Marciano Siniscalchi, the editor, and the referees for helpful comments, and David Avis, Richard McKelvey, and Ted Turocy with help on programming. The first author was supported by a Heisenberg grant from the Deutsche Forschungsgemeinschaft.

1. Introduction

In this paper we present an algorithm for computing a Nash equilibrium of a two-person game in extensive form with perfect recall. The computed equilibrium is normal form perfect. If the game has several equilibria, they can potentially be found by varying the starting point of the algorithm. The method is fast since it uses the compact "sequence form" of the extensive game (see the references below) instead of its reduced normal form. It is simple because it is a version of Lemke's algorithm for linear complementarity problems. We have implemented it in exact arithmetic, which guarantees numerical stability. Computational experiments show that the number of pivoting steps of our algorithm to find an equilibrium is of the same order as that of the simplex algorithm for linear programming applied to a comparable zero-sum game. "Typical" games with several hundred nodes are solved in less than a minute where it would be hopeless to use the reduced normal form. Our method therefore puts much more complex games in computational reach, even more so as computers get faster.

The algorithm is a synthesis of previous, partly independent work by the authors and Daphne Koller and Nimrod Megiddo. For two-person games in normal form, van den Elzen and Talman (1991, 1999) (see also van den Elzen, 1993) described a complementary pivoting algorithm that traces a piecewise linear path from a given starting vector to an equilibrium. If the starting vector is a completely mixed strategy pair, then the computed path leads to a perfect equilibrium. The free choice of the starting vector makes it possible to compute several equilibria if they exist.

This pivoting algorithm can be applied to an extensive game by converting it to its normal form. Then, the variables are probabilities for pure strategies, each of which is a combination of choices, one choice for each information set. The number of strategies therefore increases exponentially with the number of information sets. The number of information sets is typically proportional to the size of the game tree (the number of tree nodes), so then the size of the normal form is exponential. Even the reduced normal form (where strategies that differ only in choices at unreachable information sets are identified) shows an exponential-type growth in that case. For example, the games with N tree nodes studied in our computational experiments

have on the order of $2^{\sqrt{N}}$ reduced strategies rather than on the order of 2^N unreduced strategies, which nevertheless leads to an "explosion" in size. Each pivoting step updates the entire linear system derived from the payoff matrices, which is very slow for matrices of exponential size.

The sequence form of the extensive game (Romanovskii, 1962; von Stengel, 1996) is a strategic description where pure strategies are replaced by sequences of choices that lead to a node of the game tree, so there are at most as many sequences as there are nodes. The dimensions of the resulting matrix are proportional to the game tree size. Each pivoting step applied to this system is therefore *computationally efficient*. An algorithm is called computationally efficient if its asymptotic running time is bounded by a polynomial in the input size. For the overall number of pivoting steps, this is only an empirical observation. Our practical experiments show that the number of pivoting steps to find an equilibrium is about the same as the matrix dimension. The pivoting method, like the simplex algorithm for linear programming, is not polynomial in theory (certain specifically constructed worst cases take exponential time), but works well in practice.

Koller, Megiddo, and von Stengel (1996) applied the complementary pivoting algorithm by Lemke (1965) to the sequence form. As before, each pivoting step takes polynomial time, and the number of pivoting steps is empirically a polynomial in the tree size. However, this algorithm finds only one equilibrium and it is not certain whether this equilibrium is normal form perfect.

Here we show how to combine the (empirical) computational efficiency of the algorithm of Koller, Megiddo, and von Stengel (1996) and the flexibility of the algorithm of van den Elzen and Talman (1991). Our method is a variation of Lemke's algorithm and operates on the sequence form. It can be started anywhere to search for more than one equilibrium. If the starting strategy vector is completely mixed, the equilibrium found is *normal form perfect*. Equivalently, it is a Nash equilibrium in *undominated strategies* since the game has two players (van Damme, 1987).

The key to our result is the new observation that the algorithm of van den Elzen and Talman is equivalent to Lemke's algorithm for a specific auxiliary vector. This is readily applied to the sequence form, as described in Section 3 below. We

then study the nature of the computed path. The path and the equilibrium found have all properties of the normal form in a compact representation.

The implementation of our algorithm also resolves a number of technical difficulties of degeneracy and numerical accuracy. Degeneracy is intrinsic for extensive games, even with generic payoffs and when using the sequence form, since the probabilities for the players' behavior off the equilibrium path are underdetermined. In order to avoid a well-known numerical instability of Lemke's algorithm (Tomlin, 1978), we employ arbitrary precision arithmetic, and yet achieve good running times due to the use of "integer pivoting".

We also give a concise exposition of the sequence form in Section 2, and show, more explicitly than in earlier publications, how it relates to the normal form via equation (2.2). The sequence form defines an equilibrium problem where each player's strategy space is a polytope. Charnes (1953) described the solution of zero-sum games that are constrained in this way. For a game in extensive form, Romanovskii (1962) derived such a constrained matrix game which is equivalent to the sequence form. Until recently, this publication was overlooked in the English-speaking community. Eaves (1973) applied Lemke's algorithm to games which include polyhedrally constrained bimatrix games, but with different parameters than we do. Dai and Talman (1993) described an algorithm that corresponds to ours but requires simple polyhedra as strategy spaces, which is not the case for the sequence form. Selten (1988, pp. 226, 237ff) defined sequence form strategy spaces to exploit their linearity, but not for computational purposes. Recent surveys on algorithms for computing Nash equilibria are McKelvey and McLennan (1996) and von Stengel (2000).

The setup of the paper is as follows. Section 2 recalls the notion of the sequence form, its derivation from the extensive game, and how its equilibria are the solutions to a corresponding linear complementary problem. The algorithm is presented in Section 3 and illustrated in Section 4 with an example. In Section 5 we prove that the equilibrium found is normal form perfect if the starting strategy vector is completely mixed, and note that the algorithm mimics the *linear tracing procedure*. Section 6 discusses the handling of degeneracy. In Section 7 we show that our method is an instance of a *homotopy*, and mention how to find equilibria of negative index.

Section 8 compares the method with other algorithms. In Section 9, we present results of computational experiments.

2. The sequence form linear complementarity problem

We consider extensive two-person games, with conventions similar to von Stengel (1996) and Koller, Megiddo, and von Stengel (1996). An extensive game is given by a tree with a finite number of nodes, chance moves with positive probabilities, payoffs to both players at the *leaves* (the terminal nodes), and information sets partitioning the set of remaining *decision* nodes. The *choices* of a player at an information set are denoted by labels of tree edges. For simplicity, labels corresponding to different choices anywhere in the tree are distinct. On the unique path from the root to a node of the tree, the labels denoting the choices of a particular player define a *sequence* of choices for that player. We assume that both players have *perfect recall*. By definition, this means that all nodes in an information set h of a player define the same sequence σ_h of choices for that player. Under that assumption, each choice c at h is the last choice of a unique sequence $\sigma_h c$. This defines all possible sequences of a player except for the empty sequence $\emptyset$. The set of choices at an information set h is denoted C_h. The set of information sets of player i is H_i, and the set of his sequences is S_i, so

$$S_i = \{\, \emptyset \,\} \;\cup\; \{\, \sigma_h c \mid h \in H_i,\; c \in C_h \,\}.$$

The *size* of the extensive game is the amount of data needed to specify it. It is proportional to the total number of nodes of the game tree. The number $|S_i|$ of sequences of player i is $1 + \sum_{h \in H_i} |C_h|$, which is at most linear in the size of the extensive game.

A *behavior strategy* β of player i is given by probabilities $\beta(c)$ for his choices c which fulfill $\beta(c) \geq 0$ and $\sum_{c \in C_h} \beta(c) = 1$ for all h in H_i. This definition of β can be extended to the sequences σ in S_i by writing

$$\beta[\sigma] = \prod_{c \text{ in } \sigma} \beta(c). \tag{2.1}$$

A *pure strategy* π of a player is a behavior strategy with $\pi(c) \in \{0,1\}$ for all choices c. The set of pure strategies of player i is denoted P_i. Thus, $\pi[\sigma] \in \{0,1\}$ for all sequences σ in S_i. The pure strategies π with $\pi[\sigma] = 1$ are those "agreeing" with σ by prescribing all the choices in σ, and arbitrary choices at the information sets not touched by σ.

In the normal form of the extensive game, one considers pure strategies and their probability mixtures. A *mixed strategy* μ of player i assigns a probability $\mu(\pi)$ to every π in P_i. In the *sequence form* of the extensive game, one considers the sequences of a player instead of his pure strategies. A randomized strategy of player i is described by the *realization probabilities* of playing the sequences σ in S_i. For a behavior strategy β, these are obviously $\beta[\sigma]$ as in (2.1). For a mixed strategy μ of player i, they are given by

$$\mu[\sigma] = \sum_{\pi \in P_i} \pi[\sigma]\mu(\pi). \tag{2.2}$$

For player 1, this defines a map x from S_1 to $\mathbb{R}$ by $x(\sigma) = \mu[\sigma]$ for σ in S_1 which we call the *realization plan* of μ or a realization plan for player 1. A realization plan for player 2, similarly defined on S_2, is denoted y. The important properties of realization plans are stated in the following two lemmas (Koller and Megiddo, 1992; von Stengel, 1996).

LEMMA 2.1: *For player 1, x is the realization plan of a mixed strategy if and only if $x(\sigma) \geq 0$ for all $\sigma \in S_1$ and*

$$\begin{aligned} x(\emptyset) &= 1, \\ \sum_{c \in C_h} x(\sigma_h c) &= x(\sigma_h), \qquad h \in H_1. \end{aligned} \tag{2.3}$$

A realization plan y of player 2 is characterized analogously.

PROOF: Equations (2.3) hold for the realization probabilities $x(\sigma) = \beta[\sigma]$ for a behavior strategy β and thus for every pure strategy π, and therefore for their convex combinations in (2.2) with the probabilities $\mu(\pi)$. □

To simplify notation, we write realization plans as vectors $x = (x_\sigma)_{\sigma \in S_1}$ and $y = (y_\sigma)_{\sigma \in S_2}$ with sequences as subscripts. According to Lemma 2.1, these vectors

are characterized by

$$x \geq \mathbf{0}, \quad Ex = e, \qquad y \geq \mathbf{0}, \quad Fy = f \tag{2.4}$$

for suitable matrices E and F, and vectors e and f that are equal to $(1, 0 \ldots, 0)^\top$, where E and e have $1 + |H_1|$ rows and F and f have $1 + |H_2|$ rows; an example for E, e, F, and f is given in (2.6) below. Inequalities like (2.4) hold componentwise and $\mathbf{0}$ denotes a vector of zeroes. The number of information sets and therefore the number of rows of E and F is at most linear in the size of the game tree.

Mixed strategies of a player are called *realization equivalent* (Kuhn, 1953) if they define the same realization probabilities for all nodes of the tree given any strategy of the other player.

LEMMA 2.2: *Two mixed strategies μ and μ' of player i are realization equivalent if and only if they have the same realization plan, that is, $\mu[\sigma] = \mu'[\sigma]$ for all $\sigma \in S_i$.*

PROOF: Consider (2.2) as defining a linear map from $\mathbb{R}^{|P_i|}$ to $\mathbb{R}^{|S_i|}$ that maps the vector $(\mu(\pi))_{\pi \in P_i}$ to $(\mu[\sigma])_{\sigma \in S_i}$ with the fixed coefficients $\pi[\sigma]$, $\pi \in P_i$. Then mixed strategies with the same image under this map are clearly realization equivalent. □

The linear map in the preceding proof maps the simplex of mixed strategies of a player to the polytope of realization plans. These polytopes are characterized by (2.4) as asserted by Lemma 2.1. They define the player's *strategy spaces* in the sequence form and are denoted by

$$X = \{\, x \mid x \geq \mathbf{0},\ Ex = e \,\}, \qquad Y = \{\, y \mid y \geq \mathbf{0},\ Fy = f \,\}. \tag{2.5}$$

The vertices of X and Y are the players' pure strategies up to realization equivalence, which is the identification of pure strategies used in the *reduced normal form* of the game (for generic payoffs).

Figure 2.1 shows an extensive game where the choices of player 1 and player 2 are denoted by the upper and lower case letters L, R, S, T and a, b, c, d, respectively.

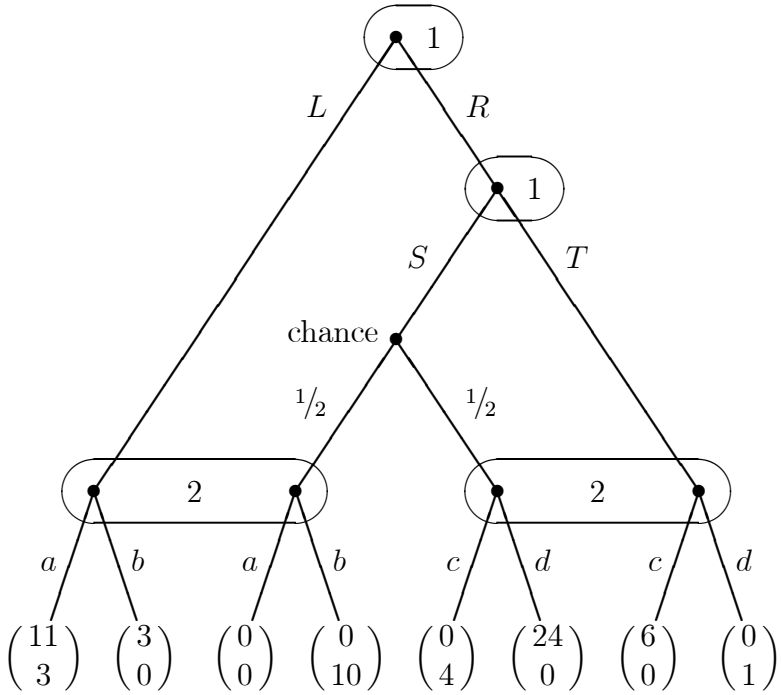

FIGURE 2.1.—A two-person extensive game.

The sets of sequences are $S_1 = \{\emptyset, L, R, RS, RT\}$ and $S_2 = \{\emptyset, a, b, c, d\}$. In the constraints (2.4) we have

$$E = \begin{bmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ & & -1 & 1 & 1 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ -1 & & & 1 & 1 \end{bmatrix}, \quad e = f = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{2.6}$$

Sequence form *payoffs* are defined for pairs of sequences whenever these lead to a leaf, multiplied by the probabilities of chance moves on the path to the leaf. This defines two sparse matrices A and B of dimension $|S_1| \times |S_2|$ for player 1 and player 2, respectively. For the game in Figure 2.1, A and B are shown in Figure 2.2. When the players use the realization plans x and y, the expected payoffs are $x^\top Ay$ for player 1 and $x^\top By$ for player 2. These terms represent the sum over all leaves of the payoffs at leaves multiplied by their realization probabilities.

Using linear programming duality, von Stengel (1996) showed that any Nash equilibrium of the game is a pair (x, y) of realization plans so that there exist vectors u, v, r, s that fulfill the linear constraints

$$A = \begin{array}{c|cccc|c} & \emptyset & a & b & c & d & \\ \hline \emptyset & & & & & & \emptyset \\ L & & 11 & 3 & & & L \\ R & & & & & & R \\ RS & & 0 & 0 & 0 & 12 & RS \\ RT & & & & 6 & 0 & RT \end{array} \qquad B = \begin{array}{c|cccc|c} & \emptyset & a & b & c & d & \\ \hline \emptyset & & & & & & \emptyset \\ L & & 3 & 0 & & & L \\ R & & & & & & R \\ RS & & 0 & 5 & 2 & 0 & RS \\ RT & & & & 0 & 1 & RT \end{array}$$

FIGURE 2.2.—Sequence form payoff matrices A and B for the game in Figure 2.1. Rows and columns correspond to the sequences of the players which are marked at the side. Any sequence pair not leading to a leaf has matrix entry zero, which is left blank.

$$\begin{aligned} x\,, \quad y &\geq \mathbf{0} \\ Ex &= e \\ Fy &= f \\ r = E^\top u - Ay &\geq \mathbf{0} \\ s = F^\top v - B^\top x &\geq \mathbf{0} \end{aligned} \tag{2.7}$$

and the complementarity condition

$$x^\top r = 0, \qquad y^\top s = 0\,. \tag{2.8}$$

The vectors u and v have dimension $1 + |H_1|$ and $1 + |H_2|$, respectively, and are unconstrained in sign. The nonnegative slack vectors r and s have dimension $|S_1|$ and $|S_2|$, respectively.

Conditions (2.7) and (2.8) define a *linear complementarity problem* or LCP. A standard LCP is specified by an $n \times n$ matrix M and an n-vector b. The problem is to find n-vectors z and w so that

$$z \geq \mathbf{0}, \qquad w = b + Mz \geq \mathbf{0}, \qquad z^\top w = 0\,. \tag{2.9}$$

The condition $z^\top w = 0$ states that the nonnegative vectors $z = (z_1, \ldots, z_n)^\top$ and $w = (w_1, \ldots, w_n)^\top$ are *complementary*, that is, at least one variable of each pair (z_i, w_i) for $1 \leq i \leq n$ is zero.

The LCP defined by (2.7) and (2.8) is a more general *mixed* LCP (see Cottle, Pang, and Stone, 1992, p. 29). Here $z = (u, v, x, y)^\top$ and $w = (\mathbf{0}, \mathbf{0}, r, s)^\top$ and certain variables z_i (the components of u and v) are unrestricted in sign and the corresponding variable w_i is always zero, so that z and w are also complementary.

3. The algorithm

Lemke (1965) described an algorithm for solving the LCP (2.9). It uses an additional n-vector d, called *covering vector*, with a corresponding scalar variable z_0, and computes with *basic solutions* to the augmented system

$$z \geq \mathbf{0}, \quad z_0 \geq 0, \qquad w = b + Mz + dz_0 \geq \mathbf{0}, \qquad z^\top w = 0\,. \tag{3.1}$$

At initialization, z_0 has a positive value. The algorithm then performs a sequence of *complementary pivoting* steps. At each step, one variable of a complementary pair (z_i, w_i) leaves and then its complement enters the basis. In a mixed LCP, a variable z_i without sign restrictions never leaves the basis. The goal is that eventually z_0 leaves the basis and then has value zero, so that the LCP is solved. Koller, Megiddo, and von Stengel (1996) give a detailed exposition of Lemke's algorithm and show that it terminates for the LCP derived from the sequence form if $d = (1, 1, \ldots, 1)^\top$.

We choose a covering vector d that is related to the starting point for our computation. Let (p, q) be an arbitrary *starting vector*, that is, a pair of realization plans for the two players, so that

$$p \geq \mathbf{0}, \quad Ep = e, \qquad q \geq \mathbf{0}, \quad Fq = f, \tag{3.2}$$

and let

$$d = \begin{bmatrix} e \\ f \\ -Aq \\ -B^\top p \end{bmatrix}. \tag{3.3}$$

We augment the mixed LCP with constraints (2.7) with d as in (3.3) and obtain analogous to (3.1)

$$
\begin{array}{rrrr}
 & x\,, & y\,, & z_0 \geq \mathbf{0} \\
 & Ex & + & e\,z_0 = e \\
 & & Fy + & f\,z_0 = f \\
r = E^\top u & & - Ay - & (Aq)z_0 \geq \mathbf{0} \\
s = & F^\top v - B^\top x & & - (B^\top p)z_0 \geq \mathbf{0}
\end{array}
\tag{3.4}
$$

and the complementarity condition (2.8). An initial solution is given by $z_0 = 1$, $x = \mathbf{0}$, $y = \mathbf{0}$, and suitable vectors u and v so that $E^\top u \geq Aq$ and $F^\top v \geq B^\top p$, that is, $r \geq \mathbf{0}$ and $s \geq \mathbf{0}$.

Conditions (3.4) and (2.8) hold for all points on the piecewise linear path computed by the algorithm. In the remainder of this section, we show that this path induces a path in the product $X \times Y$ of the two strategy spaces defined in (2.5), which begins at the starting vector (p, q) and ends at an equilibrium. The points $(\overline{x}, \overline{y})$ on this path are derived from (x, y) in (3.4) as follows.

LEMMA 3.1: *For a solution* (u, v, x, y, z_0) *to* (3.4)*, let*

$$
\overline{x} = x + pz_0, \qquad \overline{y} = y + qz_0\,. \tag{3.5}
$$

Then $\overline{x} \in X$*,* $\overline{y} \in Y$*, and* $x_\emptyset = y_\emptyset = 1 - z_0 \geq 0$*.*

PROOF: Constraints (3.4) and (3.2) imply $\overline{x} \geq \mathbf{0}$, $\overline{y} \geq \mathbf{0}$, $E\overline{x} = E(x + pz_0) = Ex + (Ep)z_0 = Ex + ez_0 = e$, and similarly $F\overline{y} = f$. By (2.3) and (2.4), the first of each of these equations reads $x_\emptyset + z_0 = 1$ and $y_\emptyset + z_0 = 1$, respectively. □

By Lemma 3.1, any solution to (3.4) fulfills $0 \leq z_0 \leq 1$. The algorithm terminates as soon as $z_0 = 0$, so that $x = \overline{x} \in X$ and $y = \overline{y} \in Y$ and (x, y) is an equilibrium. At intermittent steps of the computation with $0 < z_0 < 1$, the pair $(\overline{x}, \overline{y})$ in (3.5) can be seen as a convex combination of a pair (x^*, y^*) of realization plans and the starting pair (p, q) with weights $1 - z_0$ and z_0, respectively. Namely, let

$$
x^* = x \cdot 1/(1 - z_0), \qquad y^* = y \cdot 1/(1 - z_0), \tag{3.6}
$$

so that $\overline{x} = x + pz_0 = x^*\,(1 - z_0) + p\,z_0$ and $\overline{y} = y + qz_0 = y^*\,(1 - z_0) + q\,z_0$. By (3.4), $Ex = e(1 - z_0)$ and $Fy = f(1 - z_0)$, which implies $x^* \in X$ and $y^* \in Y$. The

positive components x_σ and y_σ of x and y are the same as the positive components of x^* and y^*, up to scalar multiplication with $1 - z_0$. By the following lemma, these represent best response sequences σ to the current pair $(\overline{x}, \overline{y})$ of realization plans.

LEMMA 3.2: *Let (u, v, x, y, z_0) be a solution to (3.4) and (2.8) with $z_0 < 1$, and let $\overline{x}$ and $\overline{y}$ be as in (3.5), and x^* and y^* as in (3.6). Then (x^*, y^*) is a pair of realization plans where x^* is a best response to $\overline{y}$ and y^* is a best response to $\overline{x}$.*

PROOF: In the following, consider $\overline{x}$ and $\overline{y}$ as given in (3.5) and x^* and y^* as in (3.6), but then allow to re-use the variables x and u. A realization plan x is a best response to $\overline{y}$ if and only if it maximizes the expected payoff $x^\top(A\overline{y})$ subject to $Ex = e$, $x \geq \mathbf{0}$. The dual of this linear program (LP) is to find u minimizing $e^\top u$ subject to $E^\top u \geq A\overline{y}$. Feasible solutions x and u to this primal-dual pair of LPs are optimal if and only if they fulfill the complementary slackness condition

$$x^\top(E^\top u - A\overline{y}) = 0\,. \tag{3.7}$$

For x and u as part of the given solution to (3.4) and (2.8), all of these conditions are fulfilled except for $Ex = e$. However, replacing x by x^* does fulfill (3.7) and $Ex^* = e$, $x^* \geq 0$ since x^* is a positive scalar multiple of x by (3.6). That is, x^* is indeed a best response to $\overline{y}$. Similarly, y^* in (3.6) is a best response to $\overline{x}$. □

In order to leave the starting vector (p, q), it is necessary to find solutions to (3.4) and (2.8) where $z_0 < 1$ is possible. This is the technical problem of a suitable *initialization* of our algorithm. Whenever z_0 decreases from one, usually *several* components of x (and similarly of y) have to become simultaneously nonzero in the equations $Ex = e(1 - z_0)$, which are the same homogeneous equations as in (2.3) except for the first, nonhomogeneous equation $x_\emptyset = 1 - z_0$ which is different. The initial solution $x = \mathbf{0}$, $y = \mathbf{0}$ does not show which components of x and y should be increased. One of these components is the first entering variable, the others must belong to the initial basis. In our implementation, we initialize Lemke's algorithm by starting it such that it automatically performs a sequence of degenerate pivoting steps that bring all components of u and v and suitable components of x and y into the basis, as shown in detail in our discussion paper (von Stengel, van den Elzen, and Talman, 1996).

For expository purposes, we explain an equivalent way of finding the initial basis by linear programming, similarly to Kamiya and Talman (1990) and Dai and Talman (1993). This initialization step is motivated by Lemma 3.2. Compute a best response x^* to q and a best response y^* to p. That is, x^* is a solution to the LP: maximize $x^\top(Aq)$ subject to $Ex = e$, $x \geq \mathbf{0}$, and y^* to the LP: maximize $(p^\top B)y$ subject to $Fy = f$, $y \geq \mathbf{0}$. This yields also corresponding optimal dual vectors u and v so that ${x^*}^\top(E^\top u - Aq) = 0$ and ${y^*}^\top(F^\top v - B^\top p) = 0$. We may assume that x^* and y^* are *basic solutions* to these two LPs, for example as they are computed by the simplex algorithm for linear programming. That is, an invertible submatrix of each matrix E and F determines the respective basic components x^*_σ and y^*_σ which may become positive, and determines uniquely u and v, respectively. Then, the basis to start Lemke's algorithm contains z_0, all components of u and v, all but one of the variables x_σ and y_σ corresponding to the basic LP variables x^*_σ and y^*_σ above (the missing one is the first entering variable), and the slack variables r_σ and s_σ in $r = E^\top u - Aq$ and $s = F^\top v - B^\top p$ for the other sequences σ. We obtain the following procedure.

ALGORITHM 3.3: *Consider an extensive game for two players with perfect recall, and its sequence form with payoff matrices A and B and constraint matrices E and F for player 1 and player 2, respectively. Choose a starting vector (p, q) fulfilling* (3.2)*. Construct the augmented mixed LCP with constraints* (3.4) *and* (2.8)*. Solve this LCP as follows.*

(a) *Find an initial basic solution with $z_0 = 1$ where the basic variables are z_0, all components of u and v, all but one of the components of x and y representing best response sequences against q and p, respectively, and slack variables r_σ and s_σ for the nonoptimal sequences σ.*

(b) *Iterate by complementary pivoting steps applied to pairs (x_σ, r_σ) or (y_σ, s_σ) of complementary variables.*

(c) *As soon as z_0 becomes zero, let z_0 leave the basis and pivot. Terminate. The computed equilibrium is (x, y).*

Lemma 3.1 shows that in the course of the computation, the values of x, y, and z_0 determine always a pair $(\overline{x}, \overline{y})$ of realization plans and thus a path in the product $X \times Y$ of the two strategy spaces. We are only interested in this path, since the basic variables in u and v are uniquely determined.

It remains to show that the algorithm *terminates*. With the above interpretation, we can exclude *ray termination*, which may cause Lemke's algorithm to fail, because the path cannot leave the strategy space. Thus, the algorithm terminates if the path is unique in the sense that no basis is revisited. This is achieved by a systematic *degeneracy resolution* which we discuss in Section 6.

4. Illustration of the algorithm

We illustrate the computation for the game in Figure 2.1. The constraints for the strategy spaces X and Y in (2.5) are given by (2.6). We denote the elements of X and Y by $\overline{x}$ and $\overline{y}$ as in (3.5). Figure 4.1 shows X for the possible values of $\overline{x}_L$, $\overline{x}_{RS}$, $\overline{x}_{RT}$. Figure 4.2 shows Y with the pairs $\overline{y}_a, \overline{y}_b$ and $\overline{y}_c, \overline{y}_d$ corresponding to the vertical and horizontal coordinates of a square, respectively, since $\overline{y}_a + \overline{y}_b = 1$ and $\overline{y}_c + \overline{y}_d = 1$. The redundant variables $\overline{x}_\emptyset$, $\overline{x}_R$, and $\overline{y}_\emptyset$ are not shown since their value is known, and they also have no payoff entry in Figure 2.2.

Each strategy space is subdivided into best response regions for the sequences of the other player. In Figure 4.2, these are the sequences L, RS, RT of player 1, which correspond to player 1's pure strategies in the reduced normal form. In Figure 4.1, X is partitioned twice, namely into regions where sequence a or b is a best response of player 2, and independently into regions where c or d is a best response. This multiple partition of X into best response regions results because a, b and c, d are the choices at *parallel* information sets h and h' where $\sigma_h = \sigma_{h'}$. This is also reflected in the structure of F and the complementary slackness condition for the constraints $F^\top v \geq B^\top \overline{x}$, as explained in detail in von Stengel et al. (1996). In effect, the four pure strategies of player 2 consisting of the choice pairs $\langle a, c\rangle$, $\langle b, c\rangle$, $\langle a, d\rangle$, and $\langle b, d\rangle$ appear both as vertices of the strategy space Y in Figure 4.2 and as intersections of best response regions partitioning X in Figure 4.1.

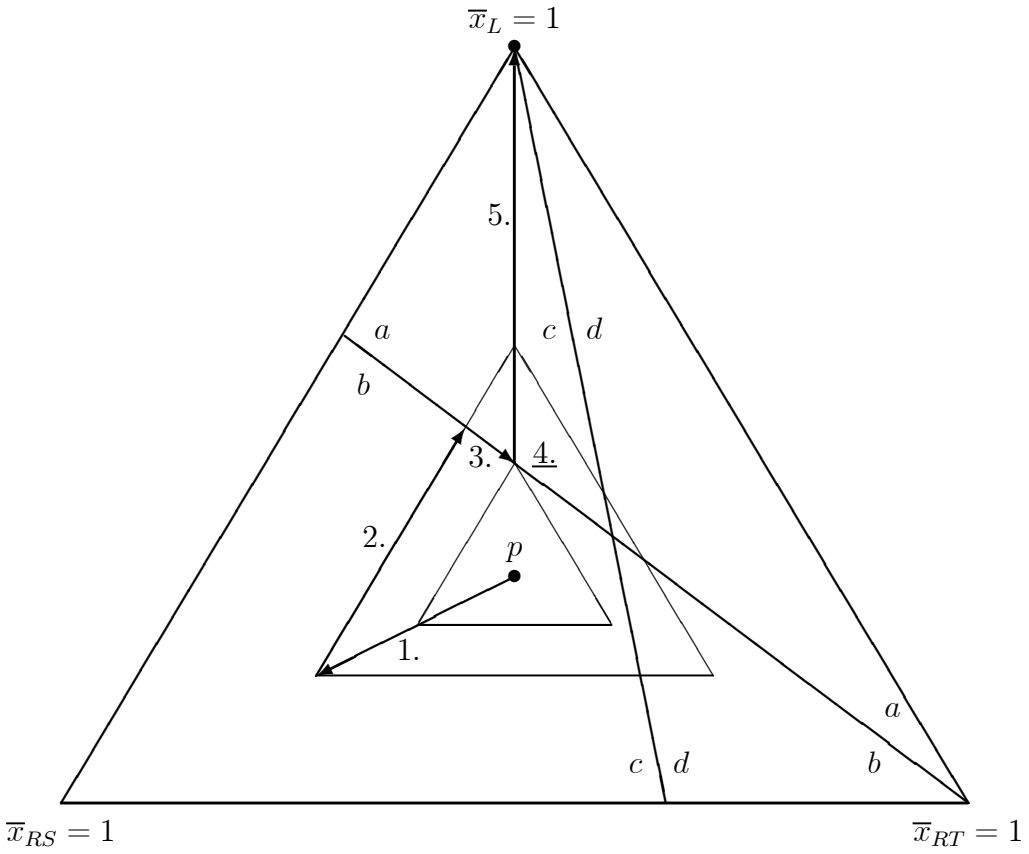

FIGURE 4.1.—Strategy space X of player 1 for the sequence form of the game in Figure 2.1, with best response sequences of player 2. Computation steps are indicated by arrows or as underlined steps with no change for player 1. The starting point p for player 1 is $(p_L, p_{RS}, p_{RT}) = (3/10, 7/20, 7/20)$.

We choose the starting vector (p, q) defined by

$$(p_L, p_{RS}, p_{RT}) = (3/10, 7/20, 7/20), \qquad (q_a, q_b, q_c, q_d) = (1/3, 2/3, 1/3, 2/3). \tag{4.1}$$

In Figure 4.1 and 4.2, p and q are marked by a dot in the interior of each strategy space. The unique best response sequence of player 1 to q is RS, and the unique best response sequences of player 2 to p are b and c. Among the components of x and y, the initial basic variables and the first entering variable in the system (3.4) are therefore x_{RS}, y_b, and y_c. The algorithm performs the following steps as indicated in the figures.

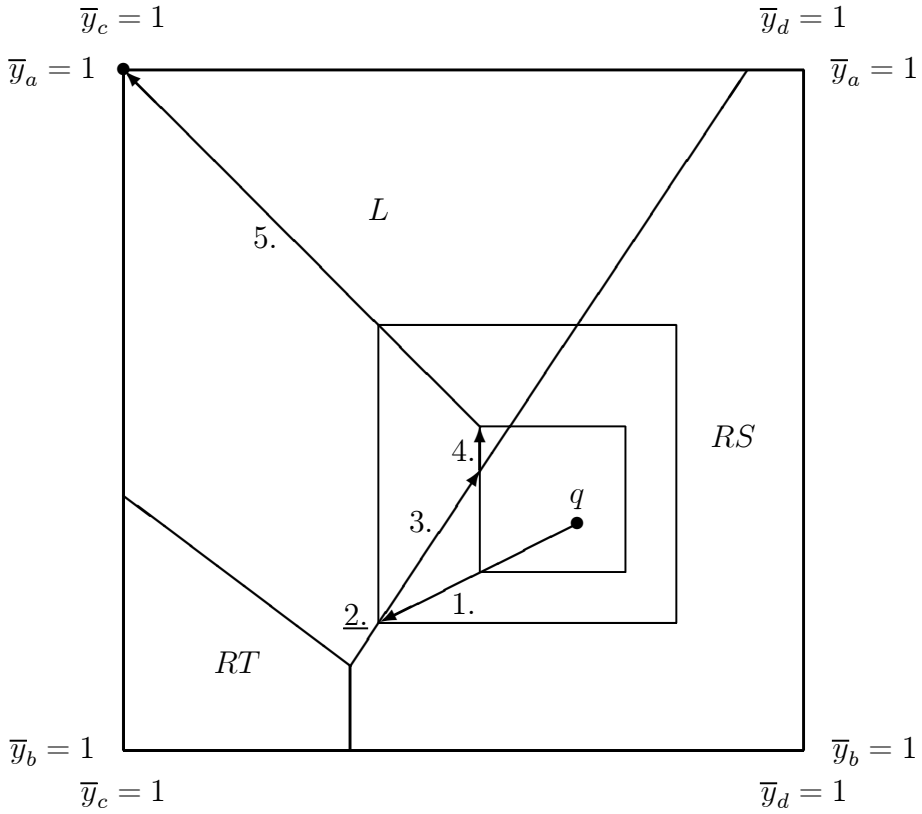

FIGURE 4.2.—Strategy space Y of player 2 for the sequence form of the game in Figure 2.1, with best responses of player 1 and computation steps. The starting point q for player 2 is $(q_a, q_b, q_c, q_d) = (1/3, 2/3, 1/3, 2/3)$.

1. The first step is the line segment starting at (p, q) so that $(\overline{x}, \overline{y})$ in (3.5) changes by decreasing z_0 from one and increasing at the same time the variables x_{RS}, y_b, y_c from zero. When $z_0 = 9/16$, the path hits the best response region for the sequence L of player 1 in Figure 4.2 because the slack r_L of the payoff for that sequence becomes zero.

For any z_0, the current pair $(\overline{x}, \overline{y})$ of realization plans defined by (3.5) belongs to $X(z_0) \times Y(z_0)$, the product of the *restricted strategy sets* defined by

$$\begin{aligned} X(z_0) &= \{\, \hat{x} \in X \mid \hat{x}_\sigma \geq p_\sigma z_0 \ \ \forall \sigma \in S_1 \,\}, \\ Y(z_0) &= \{\, \hat{y} \in Y \mid \hat{y}_\sigma \geq q_\sigma z_0 \ \ \forall \sigma \in S_2 \,\}. \end{aligned} \tag{4.2}$$

For $z_0 = 9/16$, the set $Y(z_0)$ is shown in Figure 4.2 as a square, a smaller sized replica of the strategy space Y containing the starting point q in the same relative position. The end of the arrow "1." is the lower left corner $\overline{y} = y + qz_0$ of that

square, where only the sequences b and c of player 2 have positive components y_b and y_c and $y_a = y_d = 0$. Similarly, the end of the arrow "1." in Figure 4.1 is the corner $\overline{x} = x + pz_0$ of $X(z_0)$ with $x_{RS} > 0$ and $x_L = x_{RT} = 0$.

2. Since the slack r_L has become zero, it is replaced by its complementary variable x_L that is now increased from zero, according to step (b) of Algorithm 3.3. That is, r_L has left and x_L enters the basis. When x_L is increased, then z_0 can neither decrease since this would make RS nonoptimal, nor increase since this would make L nonoptimal (see Figure 4.2). So z_0 remains unchanged. Since b and c are still the unique best responses for player 2, his current position in $Y(z_0)$ is unchanged, marked with "2." (underlined) in Figure 4.2. For player 1, the arrow "2." in Figure 4.1 denotes an increase of x_L along the boundary of $X(z_0)$ until the best response set of the sequence a of player 2 is reached. Then, the basic slack variable s_a becomes zero and is exchanged with y_a.

3. Since r_L and r_{RS} are nonbasic and zero, the next piece of the path in Figure 4.2 must belong to the best response regions for both L and RS. The relative size of y_a can only increase if z_0 is *increased*, which *shrinks* the set $Y(z_0)$. By the same shrinking factor, $X(z_0)$ becomes a smaller triangle in Figure 4.1, until x_{RS} becomes zero, which happens when z_0 is increased to $^{60}/_{77}$. Then the end of the arrow "3." points to the corner $\overline{x} = x + pz_0$ of $X(z_0)$ where x_L is the only positive component of x. The variable x_{RS} leaves the basis and is replaced by its complement r_{RS}, so that in the next step, the path leaves the best response region for RS in Figure 4.2.

4. Since y_a, y_b, y_c are all basic, z_0 remains constant and nothing changes for player 1 in Figure 4.1. By increasing r_{RS} from zero, y_a is increased and y_b decreased until it is zero at the end of the arrow "4." in Figure 4.2. Then y_b is replaced by its complement s_b.

5. The current basis contains only x_L, y_a, y_c, so the best response sequences are L for player 1 and a and c for player 2. By increasing s_b from zero, z_0 is decreased again until it is zero, reaching at the end of the arrow "5." in both figures the equilibrium $(x, y) = (\overline{x}, \overline{y})$ with $x_L = 1$ and $y_a = y_c = 1$, which terminates the algorithm.

Observe that the starting vector (p, q) is used throughout the whole computation for reference since it determines the system (3.4) and the sets $X(z_0)$ and $Y(z_0)$.

5. Perfect equilibria

Technically, our method is related to the algorithm by Koller, Megiddo, and von Stengel (1996). Conceptually, it is based on the algorithm by van den Elzen and Talman (1991) for the normal form. The mixed LCP with constraints (2.7) and (2.8) can also be used to characterize the equilibria (x, y) of a game in normal form with payoff matrices A and B. Then E and F each consist of a single row of ones and $e = f = 1$, so that the strategy spaces X and Y in (2.5) are the mixed strategy simplices. In that case, Lemke's algorithm with the covering vector d in (3.3) is equivalent to the algorithm by van den Elzen and Talman. This follows easily from Lemma 3.1, but has not been observed before.

The main game-theoretic property of the van den Elzen–Talman algorithm for the normal form is that the computed equilibrium is *perfect* whenever the starting vector is completely mixed. This result carries over to the sequence form, as follows.

Call a realization plan x for player 1 (similarly y for player 2) completely mixed if $x > \mathbf{0}$. By (2.3), x is the realization plan of the behavior strategy β defined by

$$\beta(c) = \frac{x(\sigma_h c)}{x(\sigma_h)}, \qquad c \in C_h, \quad h \in H_1\,. \tag{5.1}$$

The behavior strategy β assigns positive probability to any choice c. Regarded as a mixed strategy, β is therefore also completely mixed in the sense that every pure strategy is played with positive probability. Conversely, any completely mixed strategy defines a completely mixed realization plan by (2.2).

As shown in Lemma 2.2, the linear map defined by (2.2) maps the mixed strategy simplices to the sequence form strategy spaces X and Y. The path computed by Algorithm 3.3 is part of $X \times Y$. A suitable pre-image of this path under the linear map (2.2) yields a piecewise linear path in mixed strategies. We show this only for player 1; the consideration for player 2 is analogous. Consider two *endpoints* x^1 and x^2 in X of a line segment $[x^1, x^2]$ of the computed path. These endpoints are defined

by two successively computed bases. Let μ^1 and μ^2 be mixed strategies of player 1 that have realization plans x^1 and x^2, respectively. In the mixed strategy simplex of player 1, the line segment connecting μ^1 and μ^2 is mapped under (2.2) to $[x^1, x^2]$ since that map is linear. Thus, $[x^1, x^2]$ is indeed the image of a line segment in mixed strategies.

The particular pre-image of $[x^1, x^2]$ in the mixed strategy simplex does not matter, because mixed strategies with the same realization plans are realization equivalent and therefore payoff equivalent. A canonical choice for μ^1 and μ^2 are the corresponding behavior strategies of player 1 as in (5.1). Only the endpoints of the line segment $[x^1, x^2]$ should be translated to behavior strategies in this way, but not every point on the segment since this does not yield a line in the mixed strategy simplex if the convex combinations of μ^1 and μ^2 are not all behavior strategies.

THEOREM 5.1: *Let the starting vector (p, q) be completely mixed. Then Algorithm 3.3 computes an equilibrium that is normal form perfect.*

PROOF: Let (x^*, y^*) be the computed equilibrium. Except for its endpoint (x^*, y^*), the last line segment of the computed path consists of pairs $(x+pz_0, y+qz_0)$ of realization plans where $z_0 > 0$, due to condition 3.3(c). Therefore, these realization plans are, like p and q, completely mixed. The equilibrium (x^*, y^*) is the *limit* of these realization plans when z_0 goes to zero, and is a pair of *best responses* to these realization plans because of the complementarity condition (2.8), since x^* and y^* have the same basic components as x and y (a similar argument is made in the proof of Lemma 3.2). These properties hold also when the computed path is translated to mixed strategies as described above. According to Selten (1975, Thm. 7), they imply that the equilibrium (x^*, y^*) is perfect in the normal form. □

Any point $(\overline{x}, \overline{y})$ on the computed path is an *equilibrium* of the game with the restricted strategy sets $X(z_0)$ for player 1 and $Y(z_0)$ for player 2 in (4.2), where any nonoptimal sequence σ has the minimum probability $p_\sigma z_0$ for player 1 and $q_\sigma z_0$ for player 2. In the final computation step when z_0 goes to zero, these can be considered as *mistake* probabilities so that the equilibrium is "trembling hand" perfect. The

equilibrium is perfect for the normal form but not necessarily for the extensive form (see van Damme, 1987, p. 114).

The relative mistake probabilities for sequences are as in the starting vector (p, q), so they can be varied. The algorithm by Wilson (1992) for a game in normal form computes also a perfect equilibrium, but with mistake probabilities for pure strategies that have different orders of magnitude, according to an initially chosen order of the pure strategies.

Another game-theoretic property of our algorithm is that it mimics the *linear tracing procedure* by Harsanyi and Selten (1988), applied to the normal form of the game. Thereby, the starting vector (p, q) is the players' *prior* which the players take into account with probability z_0, whereas $1 - z_0$ is the probability for the current strategy pair (x^*, y^*) defined in (3.6). For further details see van den Elzen and Talman (1999).

6. Degeneracy resolution

The *support* of a mixed strategy is the set of pure strategies it uses with positive probability. A game is called *degenerate* if the number of pure best responses to some mixed strategy exceeds the size of its support (this is the simplest of many equivalent definitions, see von Stengel, 2000). Degeneracy can also be defined for augmented linear systems like (3.4) and for the sequence form, where it means that certain basic solutions have basic variables with value zero. Then the leaving variable in a pivoting step may be not unique and must be determined by an additional (for example lexicographic) rule that guarantees termination of the algorithm.

A bimatrix game is nondegenerate with probability one if its payoffs are *generic*, that is, drawn independently from continuous distributions. The normal form of an extensive game, however, is often degenerate even if payoffs are generic. The reason is that there may be many best response strategies specifying choices in unreached parts of the game tree. This holds also for the sequence form, even though it is less redundant than the normal form. For example, the game in Figure 2.1 has the equilibrium (x, y) in realization plans with $x_L = 1$, $y_a = y_c = 1$. Here the sequence d

of player 2 is also a best response but has probability zero, so both the slack variable s_d and its complement y_d have value zero, one of which is a basic variable. This degeneracy is due to the structure of the game tree and not due to the payoffs, since after the choice L of player 1, the second information set of player 2 with its choices c and d is unreached. In larger games, such degeneracies can also be observed at intermediate steps of the computation.

In our algorithm, degeneracy is handled by the well-known *lexicographic method* as follows (for a detailed exposition see Koller, Megiddo, and von Stengel, 1996). In a pivoting step, the leaving variable is determined by a *minimum ratio test* applied to the right hand side of the current tableau divided by the positive entries of the entering column. In a nondegenerate game, the minimum is unique. Otherwise, the set of candidates for the leaving variable is tested again by comparing the ratios for the next column of the tableau, until a unique minimum is found. Our computational experiments show that many, sometimes even all relevant tableau columns must be iteratively tested in this way. This makes it mandatory to use exact arithmetic (see Section 9 below) in order to verify the pivoting "ties" reliably. The lexicographic rule determines the leaving variable and the computed path uniquely. Hence, no basis is repeated and the algorithm terminates.

In the computation described in Section 4, the final pivoting step where z_0 leaves the basis is degenerate since the variable s_d could leave as well. According to step (c) of Algorithm 3.3, z_0 is chosen to leave the basis. According to the lexicographic rule, s_d would leave the basis, with y_d entering and then r_{RS} leaving and x_{RS} entering, and finally z_0 leaving the basis. This determines the equilibrium (x, y) with $x_L = 1$, $y_a = 1$, $y_c = 1/12$ (see von Stengel et al., 1996). Koller, Megiddo, and von Stengel (1996) showed that the lexicographic rule guarantees termination of the algorithm, even without testing before if z_0 can leave the basis. The above example shows that the extra test for z_0 can shorten the computation, which was an open question. Recall that in Algorithm 3.3(c), the variable z_0 leaves the basis as soon as possible in order to obtain a normal form perfect equilibrium by Theorem 5.1.

The lexicographic rule maintains the invariant that all computed bases are lexico-positive, so this must also hold for the initial basis. The simplest initialization

for Algorithm 3.3 is to start Lemke's algorithm in a first phase with artificial slack variables that are complementary to u and v. The components of u and v are then brought into the basis using the lexicographic rule, and never leave again. For details see von Stengel et al. (1996).

7. Homotopy and equilibria with negative index

The presented algorithm can be viewed as a *homotopy* method. The homotopy principle unifies a number of algorithms (see Garcia and Zangwill, 1981, in particular p. 368 for Lemke's algorithm). In our case, the original system (2.7), (2.8) that defines an equilibrium is relaxed to (3.4), (2.8) by admitting the extra variable z_0. The solutions to the augmented system form a one-dimensional set. With suitable lexicographic perturbation to avoid degeneracies, this set is a one-dimensional manifold, a collection of paths that do not fork. The endpoints of these paths are the equilibria of the game, with the exception of a trivial solution (for $z_0 = 1$) given by the starting vector. This is exploited algorithmically by considering first the trivial solution and then – in the usual view of a homotopy – "deforming" the system until it represents the original system (for $z_0 = 0$) with the desired solution.

The homotopy parameter z_0 is not always changed monotonically since the decrease of z_0 often stalls and may even be temporarily reversed while the path is traversed, as in step 3 in our example. On the other hand, the non-monotonicity of the homotopy makes it globally convergent and therefore superior to optimization techniques that tend to work only locally.

The endpoints of the homotopy paths have opposite *index*, an invariant of the equilibrium (see, for example, Govindan and Wilson, 1997). The equilibrium at the end of the path that begins at the starting vector has always positive index (Garcia and Zangwill, 1981, p. 54). If the game has several equilibria, any equilibrium of positive index can be found from a starting vector that is close enough (although this might not be a practical way to find all equilibria). The equilibria of *negative* index can then be found by the following modification of Algorithm 3.3. Suppose two positively indexed equilibria (x, y) and (x', y') have been traced from the starting vectors (p, q) and (p', q'), respectively. Then the homotopy system (3.4), (2.8)

induced by (p, q) has a path with (x', y') at one end and a negatively indexed equilibrium at the other end, since (x', y') is not the equilibrium connected to (p, q). Then that negatively indexed equilibrium is found by considering the system defined by (p, q) with its initial solution (x', y') and $z_0 = 0$, and then letting z_0 increase and continuing step (b) of Algorithm 3.3 until z_0 leaves again the basis with value zero.

In our example, a second positively indexed equilibrium (x', y') is given by $x' = (x_L, x_{RS}, x_{RT}) = (0, 1/3, 2/3)$ and $y' = (y_a, y_b, y_c, y_d) = (0, 1, 2/3, 1/3)$ which is found when starting from $p' = (3/10, 7/20, 7/20)$ and $q' = (1/8, 7/8, 1/3, 2/3)$, for example. Then the algorithm proceeds from the initial solution (x', y') and $z_0 = 0$ as follows.

1. The basic variables are r_L, x_{RS}, x_{RT}, s_a, y_b, y_c, y_d. The entering variable z_0 is increased to $3/8$, where the slack variable r_L becomes zero and leaves the basis.
2. The entering variable x_L is increased to $137/560$, where s_a becomes zero and leaves. No change occurs for z_0 and y.
3. The entering variable y_a is increased until $y_a = 1/8$ where $z_0 = 0$. Then z_0 leaves the basis. The algorithm terminates with the negatively indexed equilibrium $x = (5/14, 3/14, 3/7)$ and $y = (1/8, 7/8, 2/3, 1/3)$.

8. Comparison with other algorithms

A number of existing algorithms compute an equilibrium of a two-person game using the normal form. The classical algorithm by Lemke and Howson (1964) starts from a pure strategy pair where only one of the pure strategies is a best response to the other, and follows a path by complementary pivoting until the nonoptimal strategy either becomes optimal or has probability zero. Wilson (1992) extended this algorithm with a lexicographic method so that the computed equilibrium is perfect. By shifting the lexicographic order among the pure strategies and continuing the path suitably, the computed equilibrium also fulfills a variant of stability.

Wilson (1972) applied the Lemke–Howson algorithm to the normal form of an extensive game, which is stored only for the – ideally small – support of the current mixed strategies. Only the pure strategies in this support are stored explicitly as tuples of choices. These pure strategies are computed by a subroutine that works

directly on the game tree. This subroutine is called, possibly many times, for each pivoting step in order to determine the leaving variable, which is in general not part of the current support (Wilson, 1972, p. 452).

The algorithm by van den Elzen and Talman (1991) computes a perfect equilibrium when started in the interior of the strategy space. In contrast to the Lemke–Howson algorithm, its starting point can be chosen freely.

The pivoting algorithms by Lemke and Howson (1964), Wilson (1992), and van den Elzen and Talman (1991) all use the normal form of the game. Because each pivoting step updates the entire matrix which is exponentially large compared to the game tree, this becomes exceedingly slow for larger games. The algorithms of Wilson (1972, 1992) could be combined to compute a perfect equilibrium for a game in extensive form, in order to exploit the possible sparsity of mixed strategies in extensive games (see Koller and Megiddo, 1996). However, that algorithm is still slow because of a large number of subroutine calls in each pivoting step, and has other difficulties (see von Stengel, van den Elzen, and Talman, 1997). Hence, our algorithm is substantially faster than these normal form algorithms.

The algorithm by Koller, Megiddo, and von Stengel (1996) is Lemke's algorithm based on the sequence form. In comparison to that method, our algorithm has the following advantages. Our convergence proof is straightforward because the path remains in the strategy space which precludes ray termination. The proof in Koller, Megiddo, and von Stengel (1996) is very technical. Most importantly, we can freely choose the starting vector, and that choice has a clear interpretation. In consequence, our algorithm can find several equilibria if they exist. Moreover, the computed equilibrium is normal form perfect if the starting vector is completely mixed.

9. Computational experiments

We have implemented our algorithm and compared it with the algorithm by van den Elzen and Talman (1991) for the normal form, applied to the same extensive game and to the same behavior strategy that defines the starting vector. This gives

the most direct comparison between sequence form and reduced normal form. We chose a class of games where many choices in a strategy can be left unspecified since they are irrelevant (as one would typically expect), making the reduced normal form seemingly quite tractable. The computational efficiency of the sequence form therefore does not manifest itself until the game trees have several hundred nodes. Shortly thereafter, however, the reduced normal form "explodes" in size and cannot even be tested for comparison. We have investigated such large games using the sequence form only, which is still solved within several minutes. The relatively short computation times are due to the use of integer pivoting (see Shapley, 1987, and Chvàtal, 1983, p. 444). The program was written in C and run on a 400 MHz Pentium.

The games we consider are binary trees with L choices along any path from the root to a leaf, where player 1 moves I times and player 2 moves J times, $L = I + J$. The players alternate, player 1 moving first, so that $I = J = L/2$ if L is even, and $I = (L+1)/2$ and $J = (L-1)/2$ if L is odd. The game has no chance moves. At each decision node, the player has two choices and is informed about all previous choices except the immediately preceding choice by the other player. Every information set of the game (except the one containing the tree root) therefore has two nodes, and the game has no subgames. The game tree has 2^L leaves and $2^{L+1} - 1$ nodes in total. Player 1 has $(4^I + 2)/6$ and player 2 has $(4^J - 1)/3$ information sets, each with two choices. The players' payoffs are random integers between 1 and 100. Each tree depth L is studied with up to 100 different random payoffs.

The reduced normal form of such a game is substantially smaller than the unreduced normal form. Whenever a player chooses "left", say, at an information set, then all information sets following the unused choice "right" are irrelevant, which are half of the information sets at later stages. On the other hand, for any information set h of player 1, say, there is a *parallel* information set h' (that is, $\sigma_h = \sigma_{h'}$) as soon as player 2 has moved at least twice, since then player 1 is informed about the second-to-last choice of player 2. (The same holds with players interchanged.) In any reduced strategy where h and h' are relevant, all combinations of choices at h and h' must be considered, as well as combinations of subsequently

possible choices. The latter property leads to a multiplicative growth of reduced strategies. It is not hard to prove that in a game with tree depth $L = I + J$ as described above, player 1 has $2^{2^{(I-1)}}$ and player 2 has $2^{(2^J-1)}$ reduced strategies. These numbers are shown in Table 9.1.

tree depth L	3	4	5	6	7	8	9
tree leaves	8	16	32	64	128	256	512
tree nodes N	15	31	63	127	255	511	1023
move depth I player 1	2	2	3	3	4	4	5
move depth J player 2	1	2	2	3	3	4	4
strategies player 1	4	4	16	16	256	256	65536
strategies player 2	2	8	8	128	128	32768	32768
LCP dimension RNF	8	14	26	146	386	33026	98306
sequences player 1	7	7	23	23	87	87	343
constraints player 1	4	4	12	12	44	44	172
sequences player 2	3	11	11	43	43	171	171
constraints player 2	2	6	6	22	22	86	86
LCP dimension SF	16	28	52	100	196	388	772
games tested	100	100	100	100	100*	20	10
starting vectors tested	100	100	100	100	100	100	100
RNF computing time [sec]	0.001	0.003	0.017	0.71	25.4	–	–
RNF pivoting steps	6.6	8.0	10.0	14.8	20.5	–	–
SF computing time [sec]	0.003	0.017	0.142	0.89	6.0	49.8	464.3
SF pivoting steps	14.6	25.2	45.8	94.1	191.8	397.6	983.8
equilibria per game	1.2	2.0	4.4	16.6	44.2	84.3	98.9
equilibrium outcomes	1.2	1.5	2.5	4.2	7.0	13.7	22.4

TABLE 9.1.—Data for binary game tree with depth L and two-element information sets as studied in computational experiments. The observed data are averages. *RNF only tested for 20 games.

In terms of the size of the game tree, the number of reduced strategies is given as follows. When L is odd, then player 1 has twice as many reduced strategies as player 2, namely 2^{2^J} where $I = J + 1$, $L = 2J + 1$ (when L is even, the number

of strategies of player 2 is increasingly disproportionate to that of player 1, e.g. 2^{31} compared to 2^{16} when $L = 10$). Then the game tree has $N = 2^{L+1} = 2^{2J+2}$ (minus one) many nodes, so that $\sqrt{N} = 2^{J+1}$, which shows that player 1 has $2^{\sqrt{N}/2}$ many strategies. In practical terms, the exponential "explosion" happens when $J = 4$ since 2^{15} many strategies for player 2 make the reduced normal form too large to be processed with a pivoting method.

The reduced normal form (RNF) is solved by the van den Elzen–Talman algorithm. As mentioned at the beginning of Section 5, this is the same as our method except that the constraints $Ex = e$ and $Fy = f$ each consist of a single equation to define a mixed strategy. The resulting LCP dimension is shown in Table 9.1, and is about $\frac{3}{2} \cdot 2^{\sqrt{N}/2}$ for a game tree with N nodes when L is odd, otherwise about $2^{\sqrt{N/2}-1}$. For the sequence form (SF), the number of sequences and constraints (the number of equations in $Ex = e$ and $Fy = f$, which is the number of the player's information sets plus one) give an LCP dimension of about $\frac{3}{4}N$.

Each game is solved with 100 different starting vectors. One of these is the "centroid", that is, the behavior strategy where all choices have equal probability. The other starting vectors are behavior strategies with a random behavior at each information set, which is a probability vector chosen from the uniform distribution on the respective unit simplex. Here, each information set has only two choices, so the probability for one choice is just uniformly chosen from the unit interval, and the other choice gets the complementary probability. (It is less straightforward but possible to implement a uniform distribution on the respective higher-dimensional simplex when an information set has more than two choices.) For game trees of depth $L \leq 6$, each of the 100 games resulting from different payoffs is tested with 100 different starting vectors, so that running times are averaged over 10,000 computations. For $L = 7$, 8, and 9, the sequence form is applied to 100, 20, and 10 different games, respectively, each with 100 starting vectors. The number of games does not seem to be critical – already 10 games for each level lead to very similar computation times. For $L = 7$, the normal form computation takes too long and is applied only to 20 games, that is, 2,000 times in total. For $L \geq 8$, the normal form is too large even to store the LCP matrix.

The computed equilibria for the reduced normal form and for the sequence form almost always agree, as predicted in Section 5. A rare exception are games where the lexicographic rule, which depends on the arbitrary order of LCP variables, resolves a "bifurcation" of the computed path differently, for example when two choices lead to equal payoffs.

The reduced normal form requires much fewer pivoting steps (see Table 9.1) since one step changes a strategy and thus several choices at a time. In contrast, the sequence form requires a larger number of pivoting steps (roughly the same as the LCP dimension) since these change only one choice at a time. Curiously, the number of pivoting steps is always *odd*. We have not yet explained this observation, which is not true for the Lemke–Howson algorithm, for example.

Computation times for the sequence form break about even with the reduced normal form when $L = 6$. Then, the larger number of pivoting steps for the sequence form is outbalanced by the smaller time needed to perform a pivoting step in a smaller and sparser tableau. Computation times for the reduced normal form are also more variable. The longest computation for $L = 6$ took 15.0 seconds (average 0.71, standard deviation 0.68) compared to maximally 5.4 seconds for the sequence form (average 0.89, standard deviation 0.41).

Table 9.1 shows that doubling the size N of the game tree incurs an approximately eightfold increase of the computation time for the sequence form (the small number of levels investigated allows only for a rough estimate in this regard). Empirically, the running time is therefore cubic in the input size (proportional to N^3 for a game tree with N nodes). Since the number of pivoting steps seems to be proportional to N, each step takes time proportional to N^2. The tableau itself is sparse, but the numbers involved have a larger number of digits with increasing N which slows the computation down. This is due to the use of arbitrary precision integer arithmetic, which is necessary since Lemke's algorithm with standard floating point arithmetic is numerically unstable (Tomlin, 1978). For comparison, the simplex algorithm for linear programming applied to a zero-sum game requires in practice also proportional to N many pivoting steps for an LP with N constraints (Chvàtal 1983, p. 45). The only difference to our method is that good implemen-

tations of the simplex method are numerically stable even with limited-precision floating point arithmetic, and therefore faster for larger zero-sum games.

As the games increase in size, a larger number of equilibria is found when varying the starting vector (for $L = 8$ and $L = 9$, almost every different starting vector leads to a different equilibrium, so that one should try more than 100 starting vectors here). As soon as a player can move several times during play, many of these equilibria differ only in choices away from the equilibrium path, as demonstrated by the last row in Table 9.1 that shows the number of distinct equilibrium outcomes, that is, distributions on tree leaves induced by equilibria. The support of an equilibrium outcome tends to be small. Most equilibria are in pure strategies or involve only very few information sets where a player mixes his moves. This observation – without analyzing it further – holds presumably because the game has random payoffs. At the tree sizes where the sequence form becomes relevant, it is hopeless to enumerate all equilibria since enumeration is exponential in the LCP dimension (see also Gilboa and Zemel, 1989). However, we have not tried to find as many equilibria as possible, or to find negatively indexed equilibria according to Section 7.

Authors' addresses:

B. von Stengel: Department of Mathematics, London School of Economics, Houghton St, London WC2A 2AE, United Kingdom. Email: stengel@maths.lse.ac.uk

A. H. van den Elzen and A. J. J. Talman: Department of Econometrics, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands.
Email: elzen@kub.nl, talman@kub.nl

References

A. Charnes (1953), Constrained games and linear programming. Proc. National Academy of Sciences of the U.S.A. 39, 639–641.

V. Chvátal (1983), Linear Programming. Freeman, New York.

R. W. Cottle, J.-S. Pang, and R. E. Stone (1992), The Linear Complementarity Problem. Academic Press, San Diego.

Y. Dai and A. J. J. Talman (1993), Linear stationary point problems on unbounded polyhedra. Mathematics of Operations Research 18, 635–644.

E. van Damme (1987), Stability and Perfection of Nash Equilibria. Springer, Berlin.

B. C. Eaves (1973), Polymatrix games with joint constraints. SIAM J. Appl. Math. 24, 418–423.

A. H. van den Elzen (1993), Adjustment Processes for Exchange Economies and Noncooperative Games. Lecture Notes in Economics and Mathematical Systems 402, Springer, Berlin.

A. H. van den Elzen and A. J. J. Talman (1991), A procedure for finding Nash equilibria in bi-matrix games. ZOR – Methods and Models of Operations Research 35, 27–43.

A. H. van den Elzen and A. J. J. Talman (1999), An algorithmic approach toward the tracing procedure for bi-matrix games. Games and Economic Behavior 28, 130–145.

C. B. Garcia and W. I. Zangwill (1981), Pathways to Solutions, Fixed Points, and Equilibria. Prentice-Hall, Englewood Cliffs.

I. Gilboa and E. Zemel (1989), Nash and correlated equilibria: some complexity considerations. Games and Economic Behavior 1, 80–93.

S. Govindan and R. Wilson (1997), Equivalence and invariance of the index and degree of Nash equilibria. Games and Economic Behavior 21, 56–61.

J. C. Harsanyi and R. Selten (1988), A General Theory of Equilibrium Selection in Games. MIT Press, Cambridge.

K. Kamiya and A. J. J. Talman (1990), Linear stationary point problems. CentER Discussion paper No. 9022, Tilburg University.

D. Koller and N. Megiddo (1992), The complexity of two-person zero-sum games in extensive form. Games and Economic Behavior 4, 528–552.

D. Koller and N. Megiddo (1996), Finding mixed strategies with small supports in extensive form games. International Journal of Game Theory 25, 73–92.

D. Koller, N. Megiddo, and B. von Stengel (1996), Efficient computation of equilibria for extensive two-person games. Games and Economic Behavior 14, 247–259.

H. W. Kuhn (1953), Extensive games and the problem of information. In: Contributions to the Theory of Games II, eds. H. W. Kuhn and A. W. Tucker, Annals of Mathematics Studies 28, Princeton Univ. Press, Princeton, pp. 193–216.

C. E. Lemke (1965), Bimatrix equilibrium points and mathematical programming. Management Science 11, 681–689.

C. E. Lemke and J. T. Howson, Jr. (1964), Equilibrium points of bimatrix games. Journal of the Society for Industrial and Applied Mathematics 12, 413–423.

R. D. McKelvey and A. McLennan (1996), Computation of equilibria in finite games. In: Handbook of Computational Economics, Vol. I, eds. H. M. Amman, D. A. Kendrick, and J. Rust, Elsevier, Amsterdam, pp. 87–142.

I. V. Romanovskii (1962), Reduction of a game with complete memory to a matrix game. Soviet Mathematics 3, 678–681 (Russian original: Doklady Akademii Nauk SSSR 144, 62–64).

R. Selten (1975), Reexamination of the perfectness concept for equilibrium points in extensive games. International Journal of Game Theory 4, 25–55.

R. Selten (1988), Evolutionary stability in extensive two-person games – correction and further development. Mathematical Social Sciences 16, 223–266.

L. S. Shapley (1987), The Lemke–Howson algorithm for bimatrix games, A4: How to pivot with a matrix of integers. Handout #5, lecture notes for Math 147/1, UCLA.

B. von Stengel (1996), Efficient computation of behavior strategies. Games and Economic Behavior 14, 220–246.

B. von Stengel (2000), Computing equilibria for two-person games. To appear in the Handbook of Game Theory, Vol. 3, eds. R. J. Aumann and S. Hart, North-Holland, Amsterdam. (Technical Report #253, Dept. of Computer Science, ETH Zürich.)

B. von Stengel, A. H. van den Elzen, and A. J. J. Talman (1996), Tracing equilibria in extensive games by complementary pivoting. CentER Discussion paper No. 9686, Tilburg University.

B. von Stengel, A. H. van den Elzen, and A. J. J. Talman (1997), Computing normal form perfect equilibria for extensive two-person games. Discussion paper FEW 752, Tilburg University.

J. A. Tomlin (1978), Robust implementation of Lemke's method for the linear complementarity problem. Mathematical Programming Study 7: Complementarity and Fixed Point Problems, 55–60.

R. Wilson (1972), Computing equilibria of two-person games from the extensive form. Management Science 18, 448–460.

R. Wilson (1992), Computing simply stable equilibria. Econometrica 60, 1039–1070.