COMPUTATION OF EQUILIBRIA IN FINITE GAMES*

Richard D. McKelvey
Division of Humanities & Social Sciences
California Institute of Technology
Pasadena, California 91125
rdm@hss.caltech.edu

Andrew McLennan
Department of Economics
University of Minnesota
Minneapolis, Minnesota 55455
mclennan@walleye.econ.umn.edu

July 14, 1994
June 30, 1996

*This research was funded in part by National Science Foundation grants SBR-9308637 to the California Institute of Technology and SBR-9308862 to the University of Minnesota. We are grateful to Robert Wilson and an anonymous referee for detailed comments on earlier drafts.

COMPUTATION OF EQUILIBRIA IN FINITE GAMES

Richard D. McKelvey   Andrew McLennan

Abstract

We review the current state of the art of methods for numerical computation of Nash equilibria for finite n-person games. Classical path following methods, such as the Lemke-Howson algorithm for two person games, and Scarf-type fixed point algorithms for n-person games provide globally convergent methods for finding a *sample equilibrium*. For large problems, methods which are not globally convergent, such as sequential linear complementarity methods may be preferred on the grounds of speed. None of these methods are capable of characterizing the entire set of Nash equilibria. More computationally intensive methods, which derive from the theory of semi-algebraic sets are required for finding all equilibria. These methods can also be applied to compute various equilibrium refinements.

JEL classification numbers:

Key words:

COMPUTATION OF EQUILIBRIA IN FINITE GAMES

Richard D. McKelvey   Andrew McLennan

Contents

1 Introduction **1**

2 Notation and Problem Statement **2**

3 Computing a Sample Equilibrium **5**
3.1 Two-Person Games: The Lemke-Howson Algorithm 5
3.1.1 Finding one Nash equilibrium 5
3.1.2 Finding multiple Nash equilibria 12
3.1.3 Zero sum games 13
3.1.4 Summary 14
3.2 N-Person Games: Simplicial Subdivision 14
3.2.1 Fixed triangulation 14
3.2.2 Refining the triangulation 17
3.2.3 Computational complexity 18
3.3 Non Globally Convergent Methods 18
3.3.1 Nash equilibrium as non-linear complementarity problem 18
3.3.2 Nash equilibrium as a minimum of a function 19

4 Extensive Form Games **20**
4.1 Notation 21
4.2 Extensive versus normal form 24
4.3 Computing sequential equilibria 25

5 Equilbrium Refinements **26**
5.1 Two Person Games 26
5.2 N Person Games 28

6 Finding all Equilibria **28**
6.1 Feasibility 30
6.2 Exemplary Algorithms for semi-algebraic sets 31
6.2.1 The resultant 31
6.2.2 Univariate Systems 34
6.2.3 Numerical computation of the solutions 39
6.3 Complexity of finding game theoretic equilibria 39
6.3.1 Two Person Games 41
6.3.2 N-Person Games 42

7 Practical Computational Issues **43**
7.1 Software 43
7.1.1 Current version of GAMBIT 43
7.1.2 Revisions to GAMBIT 44
7.2 Computational Complexity 47

8 References **49**

1 Introduction

In this paper, we review the current state of the art of methods for numerical computation of Nash equilibria – and refinements of Nash equilibria – for general finite n-person games. For the most part, we simply survey existing literature. However, we also provide proofs and technical details for certain results that may be well known to practitioners, but for which there is not an accessible treatment in the literature.

Although our perspective will emphasize the concerns of economists, we exclude from consideration algorithms that efficiently solve the specific games that arise in various areas of application (e.g., auctions, bargaining, matching, repeated games, stochastic games, games of perfect information.) Such specialized procedures constitute a subject area that is, at least potentially, almost as rich as the whole of economic theory. We also do not attempt to describe the very extensive literature concerned with procedures, such as programs for playing chess, that attempt to find relatively effective choices of actions without completely solving the game. Finally, although some of the algorithms we present have been implemented, so that we will be able to discuss some applications, the bulk of the work to date has been theoretical, and it is this aspect that will be emphasized.

Students of game theory, their teachers, and researchers who use these concepts, are generally aware that solving for Nash equilibria can be a tedious, error-prone affair, even when the game is very simple, and they also know that the need to solve a game arises with fair frequency. We will therefore not argue for the general utility of such software in any detail. It is, perhaps, less obvious that suitable software could support styles of research that are currently infeasible. For theorists it could be useful to test an hypothesis by systematically searching for a counterexample before launching an effort to prove it, and in some cases such a search could itself constitute a proof. Experimentalists might wish to search parameter spaces in order to obtain experimental designs that maximize the statistical distinction between competing hypotheses. Mechanism designers might conduct such searches with different goals in mind. Finally, econometric analysis of strategic decisions requires the ability to solve a given extensive or normal form repeatedly with different parameter values, for the purpose of computation of likelihood functions and subsequent parameter estimation.

The appropriate method for computing Nash equilibria for a game depends on a number of factors. The first and most important factor involves whether we want to simply find one equilibrium (a **sample equilibrium**) or find all equilibria. The problem of finding *one* equilibrium is a well studied problem, and there exist a number of different methods for numerically computing a sample equilibrium. The problem of finding all equilibria has been addressed more recently. While there exist methods for computation of all equilibria, they are very computationally intensive. With current methods, they are only feasible on small problems. We discuss those methods in this paper.

The second factor of importance concerns whether n, the number of players, is greater than two. The polynomials that arise in the definition of Nash equilibrium are of degree

$n-1$ in the variables describing the agents' mixed strategies, so for games with two players, a Nash equilibrium solves a system of linear equations over the variables in the support of the equilibrium. Among other things, if the input data are rational numbers, the set of all Nash equilibria is the union of a set of convex polyhedra, each of which can be completely characterized by specifying a finite number of extreme points (which also have rational coordinates). Because of this, for two person games there exist methods for finding exact sample Nash equilibria, and for characterizing exactly the entire set of Nash equilibria. For games with more than two players, even if the input data are rational, the set of Nash equilibria with a given support need no longer be a convex, or even connected set. Even if it is a singleton, it need not have rational coordinates. The methods that work for two person games can not typically be directly extended for n-person games.

The third factor that determines the choice of method concerns the *type* of Nash equilibria we wish to find. It is well known that not all Nash equilibria are equally attractive. For example, Nash equilibria can be dominated, and if there are multiple equilibria, they may be Pareto ranked. A large literature exists on equilibrium refinements, which defines criteria for selecting among multiple equilibria (such as perfect equilibria, proper equilibria, sequential equilibria, stable sets, etc.) The issue of equilibrium refinements has not been extensively addressed in the literature on computation of Nash equilibria. We discuss here the limited results that are available. The methods for finding a sample equilibrium are only guaranteed to find a Nash equilibrium. Thus there is no guarantee that the equilibrium found will satisfy whatever refinement condition is deemed important. So any method intended to find a sample Nash equilibrium needs to be modified to find a particular refinement. Since the set of refinements is a subset of the set of all Nash equilibria, a method that finds all Nash equilibria can serve as a basis for a method to find the set of all refined Nash equilibria, as long as we can characterize the set of refined Nash equilibria as a subset of the set of Nash equilibria in a computable way. The Tarski-Seidenberg theorem implies that most of the equilibrium refinements that have been proposed can be expressed as semi-algebraic sets. Thus, the same methods as are used to find all Nash equilibria can in principle be used to find any of these refinements.

The remainder of the paper is organized as follows. Section 2 introduces notation and states the problem. Section 3 reviews methods for computing sample equilibria in normal form games. Section 4 deals with computation of equilibria on extensive form games. Section 5 discusses the computation of equilibrium refinements. Section 6 discusses methods for finding all equilibria. Finally, Section 7 discusses practical computational issues and experience.

2 Notation and Problem Statement

Consider a finite n-person game in normal form: There is a set $N = \{1, \ldots, n\}$ of **players**, and for each player $i \in N$ a **strategy set** $S_i = \{s_{i1}, \ldots, s_{im_i}\}$, consisting of m_i **pure strategies**. For each $i \in N$, we are given a **payoff function**, $u_i : S \mapsto \mathbb{R}$, where

$S = \prod_{i \in N} S_i$.

Let $\mathcal{P}_i$ be the set of real valued functions on S_i. For elements $p_i \in \mathcal{P}_i$ we use the notation $p_{ij} = p_i(s_{ij})$. Let $\mathcal{P} = \prod_{i \epsilon N} \mathcal{P}_i$ and let $m = \sum_{i \epsilon N} m_i$. Then $\mathcal{P}$ is isomorphic to $\mathbb{R}^m$. We denote points in $\mathcal{P}$ by $p = (p_1, \ldots, p_n)$, where $p_i = (p_{i1}, \ldots, p_{im_i}) \in \mathcal{P}_i$. If $p \in \mathcal{P}$, and $p'_i \in \mathcal{P}_i$, we use the notation (p'_i, p_{-i}) for the element $q \in \mathcal{P}$ satisfying $q_i = p'_i$, and $q_j = p_j$ for $j \neq i$. We use similar notation for any vector.

The payoff function u is extended to have domain $\mathbb{R}^m$ by the rule

$$u_i(p) = \sum_{s \in S} p(s) u_i(s), \tag{2.1}$$

where we define

$$p(s) = \prod_{i \in N} p_i(s_i). \tag{2.2}$$

Let $\Delta_i = \{p_i \in \mathcal{P}_i : \sum_j p_{ij} = 1, p_i \geq 0\}$ be the set of probability measures on S_i. Let $\Delta = \prod_{i \epsilon N} \Delta_i \subseteq \mathbb{R}^m$. We use the abusive notation s_{ij} to denote the element $p_i \in \Delta_i$ with $p_{ij} = 1$. Hence, the notation (s_{ij}, p_{-i}) represents the strategy where i adopts the pure strategy s_{ij}, and all other players adopt their components of p.

Definition 1 *We say $p^* \in \mathcal{P}$ is a* **Nash equilibrium** *if $p^* \in \Delta$ and for all $i \in N$, and all $p_i \in \Delta_i$, $u_i(p_i, p^*_{-i}) \leq u_i(p^*)$.*

We start by giving several alternative characterizations of a Nash equilibrium. We first define three functions x, z, and $g : \mathcal{P} \mapsto \mathbb{R}^m$, derived from the normal form game u. For any $p \in \mathcal{P}$, and $i \in N$, and $s_{ij} \in S_i$, define the i, j^{th} component by

$$x_{ij}(p) = u_i(s_{ij}, p_{-i}) \tag{2.3}$$

$$z_{ij}(p) = x_{ij}(p) - u_i(p) \tag{2.4}$$

$$g_{ij}(p) = \max[z_{ij}(p), 0] \tag{2.5}$$

Nash equilibrium as a fixed point of a correspondence

Define the **best response correspondence** $\lambda : \Delta \mapsto\mapsto \Delta$ by

$$\lambda(p) = \arg\max_{q \in \Delta} [\sum_{i \in N} u_i(q_i, p_{-i})] \tag{2.6}$$

Then $p^* \in \Delta$ is a Nash equilibrium if and only if it is a fixed point of λ, in other words $p^* \in \lambda(p^*)$.

It is well known (and easy to prove) that λ has a closed graph, is nonempty, and convex valued. It follows by the Kakutani fixed point theorem that there is a fixed point.

Nash equilibrium as a fixed point of a function

Define $y : \Delta \mapsto \Delta$, with i, j^{th} component

$$y_{ij}(p) = \frac{p_{ij} + g_{ij}(p)}{1 + \sum_j g_{ij}(p)}. \tag{2.7}$$

Then $p^* \in \Delta$ is a Nash equilibrium if and only if it is a fixed point of y, in other words $p^* = y(p^*)$.

Since y is a continuous function from a compact set Δ into itself, it follows from the Brouwer fixed point theorem that y has a fixed point. This argument is used by Nash [1951] to prove existence of equilibrium for finite n-person games.

Nash equilibrium as a solution to a non-linear complementarity problem

The function $z : \mathcal{P} \mapsto \mathbb{R}^m$ defined above satisfies $p_i \cdot z_i(p) = 0$ for all p and each i (i. e., p_i and $z_i(p)$ are orthogonal.) A point $p^* \in \Delta$ is a Nash equilibrium if and only if $z(p^*) \leq 0$. The Nonlinear Complementarity Problem (NCP) on Δ then consists of finding a point $p \in \Delta$ with $z(p) \leq 0$. Such a point p is complementary to $z(p)$. I. e., $p_{ij} \cdot z_{ij}(p) = 0$ for all i, j. Thus the set of Nash equilibria are the solution to a NCP on Δ.

Nash equilibrium as a stationary point problem

A point $p^* \in \Delta$ is a Nash equilibrium if and only if it satisfies

$$(p_i - p_i^*) \cdot x_i(p^*) \leq 0 \tag{2.8}$$

for all i and $p \in \Delta$. This is the stationary point problem for the function $x : \Delta \mapsto \mathbb{R}^m$ on the polytope $\Delta \subseteq \mathbb{R}^m$.

Nash equilibrium as a minimum of a function on a polytope

Define the real valued function $v : \Delta \mapsto \mathbb{R}$ by

$$v(p) = \sum_{i \in N} \sum_{1 \leq j \leq m_i} [g_{ij}(p)]^2. \tag{2.9}$$

This is a continuous, differentiable real valued function satisfying $v(p) \geq 0$ for all p. Further, p^* is a Nash equilibrium if and only if it is a global minimum of v. In other words $v(p^*) = 0$.

Nash equilibria as a semi-algebraic set

The set of Nash equilibria is the set of points $p \in \mathbb{R}^n$ satisfying

$$p \in \Delta \text{ and } z(p) \leq 0. \tag{2.10}$$

But Δ is defined by a set of linear inequalities, and $z : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a polynomial in p. Hence the set of Nash equilibria is a semi-algebraic set.

From the above alternative formulations of a Nash equilibrium, it is clear that the problem of finding a Nash equilibrium for a finite game can be expressed in terms of a number of standard problems in the theory of optimization. Each of these problems has been extensively studied, although not always with the application of computing Nash equilibria as the goal. However, an array of methods can potentially be brought to bear on the problem of computing a Nash equilibrium of a finite game.

3 Computing a Sample Equilibrium

Most of the literature on computation of Nash equilibria deals with the computation of a sample Nash equilibrium. In this section we review this literature. We first discuss methods which are globally convergent. As discussed in the introduction, different methods must be used for two person and n-person games. We conclude with a brief discussion of methods that are not globally convergent.

3.1 Two-Person Games: The Lemke-Howson Algorithm

Any review of methods of computation for game theory must start with the work of Lemke and Howson [1964]. Historically, the Lemke-Howson algorithm was the first of the so called **path following algorithms**. Lemke and Howson's algorithm was developed originally for two person games and was then extended to solve more general linear complementarity problems (Lemke [1965], Eaves [1971]), of which a two person game is a specific example. Shapley [1974] describes a nice geometrical interpretation of the Lemke-Howson algorithm for the nondegenerate case, which lends itself to easy visualization if the number of strategies for both players are small enough. In this section, we give a precise description of the Lemke-Howson algorithm as modified by Eaves [1971] to deal with degenerate problems.

3.1.1 Finding one Nash equilibrium

Consider a two person game with strategy sets $S_i = \{s_{i1}, \ldots, s_{im_i}\}$ for player i, $i = 1, 2$. Let U_i be the payoff matrix for player i, with rows indexed by the strategies for player i and columns indexed by the strategies for player $\ell \neq i$. In other words, the entry in row j, column k of U_i is $u_i(s_{ij}, s_{-i,k})$. We assume, without loss of generality, that all entries in U_i are positive.

A Nash equilibrium is a pair of column vectors $p_i \in \mathbb{R}^{m_i}$ satisfying

$$U_i \cdot p_{-i} + r_i = v_i \cdot \mathbf{1} \qquad (3.1)$$

subject to $p_i \geq 0$, $r_i \geq 0$, $p_i \cdot \mathbf{1} = 1$, and $p_i \cdot r_i = 0$ for $i = 1, 2$. Here, v_i is a scalar representing player i's payoff, and $r_i \in \mathbb{R}^{m_i}$ is a column vector whose elements r_{ij} are "slack" variables which must be 0 if i uses strategy s_{ij} with positive probability. The notation $\mathbf{1}$ indicates a column vector of 1's of appropriate dimension.

If all entries in the U_i are positive, then we are assured that the v_i are positive, and we can reformulate the above problem. Define $p'_i = p_i / v_{-i}$, and $r'_i = r_i / v_i$. Then equation (3.1) can be re-written as

$$U_i \cdot p'_{-i} + r'_i = \mathbf{1} \tag{3.2}$$

subject to $p'_i \geq 0$, $r'_i \geq 0$, $p'_i \cdot \mathbf{1} = 1/v_{-i}$, and $p'_i \cdot r'_i = 0$ for $i = 1, 2$. We can drop the constraint that $p'_i \cdot \mathbf{1} = 1/v_{-i}$ at the expense of obtaining one additional solution in which $p'_1 = p'_2 = 0$. This solution is called the **extraneous solution**.

Define U to be the $m \times m$ matrix

$$U = \begin{bmatrix} 0 & U_1 \\ U_2 & 0 \end{bmatrix}, \tag{3.3}$$

and x, y, and q to be column vectors of length m:

$$x = \begin{bmatrix} r'_1 \\ r'_2 \end{bmatrix}, y = \begin{bmatrix} p'_1 \\ p'_2 \end{bmatrix}, q = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}. \tag{3.4}$$

Then equation (3.2) can be rewritten as

$$Uy + x = q. \tag{3.5}$$

subject to $x \geq 0$, $y \geq 0$, and $x \cdot y = 0$. This is exactly in the form of a **linear complementarity problem**. We now discuss methods of solving such problems.

A pair of vectors $x, y \in \mathbb{R}^m$ is said to be a **solution** of (3.5) if $Uy + x = q$. A solution is **feasible** if $x \geq 0$ and $y \geq 0$. As the intersection of finitely many half spaces of the form $\{\, \xi \in \mathbb{R}^{2m} : \nu \cdot \xi \leq c \,\}$, the set of feasible solutions is a closed convex polyhedron, which may be either empty or nonempty, and if nonempty, either bounded or unbounded. In the case of interest to us U has nonnegative entries, and each of its columns contains a positive entry, while q has positive components. These conditions guarantee that the set of feasible solutions is both nonempty (since the extraneous solution is feasible) and bounded.

A solution is said to be **complementary** if $x \cdot y = 0$. Note that a feasible solution is complementary if and only if, for each $1 \leq i \leq m$, either $x_i = 0$ or $y_i = 0$. Clearly the extraneous solution ($x = q, y = 0$) is a complementary feasible solution. Any Nash equilibrium corresponds to a complementary feasible solution. Further, any complementary feasible solution corresponds either to a Nash equilibrium or the extraneous solution.

Described geometrically, the Lemke-Howson algorithm starts at a given complementary feasible solution, then proceeds, from vertex to vertex, along a certain path of one dimensional faces of the polyhedron of feasible solutions, until it reaches a different complementary feasible solution. For "nondegenerate" problems (which are generic in the space of parameters U and q) the specific construction of the path is not difficult to describe, as we will see below. But for exceptional problems (which can easily arise in games derived from extensive forms) there are ambiguities that must be resolved.

We write the system (3.5) in the form

$$[\,U \quad I_m\,] \cdot \begin{bmatrix} y \\ x \end{bmatrix} = q. \tag{3.6}$$

Let $A = [\,U \quad I_m\,]$, and let $z = \begin{bmatrix} y \\ x \end{bmatrix}$. Consider a collection of indices $\beta = \{b_1, \ldots, b_m\}$ with $1 \leq b_i \leq 2m$, and let B^β be the $m \times m$ matrix whose i^{th} column is the ${\beta_i}^{\text{th}}$ column of A. We say that β is a **basis** if B^β is nonsingular, and we say that a solution z is a **basic solution** if there is a basis β such that $z_i = 0$ for all $i \notin \beta$. Note that for each basis β there is exactly one solution z: the components of z with indices in β are the components of $(B^\beta)^{-1}q$. A basis is **feasible** if its corresponding solution is feasible. Basic feasible solutions are vertices of the polyhedron of feasible solutions, and conversely.

A basis β is **complementary** if, for each $1 \leq j \leq m$, exactly one of j and $m + j$ is in β. A basis is i-**almost complementary** if, for all $1 \leq j \leq m$ different from i, either $j \notin \beta$ or $m + j \notin \beta$. Note that a complementary basis is i-almost complementary for every i. For an i-almost complementary basis β that is not complementary there is some $1 \leq j \leq m$ with $j \notin \beta$ and $m + j \notin \beta$; we call j the **omitted index** of β. The idea will be to start at a complementary feasible solution, then, for some i, move along a sequence of i-almost complementary feasible solutions until we reach another complementary solution.

For a basis β let $q^\beta = (B^\beta)^{-1}q$ and let $A^\beta = (B^\beta)^{-1}A$. Then define

$$T^\beta = [\,-q^\beta \quad A^\beta\,] = (B^\beta)^{-1}\,[\,-q \quad A\,]\,. \tag{3.7}$$

Let $\gamma = \{g_1, \ldots, g_m\}$ be another basis. We are particularly interested in the case in which β and γ differ by one element, so assume that $\gamma = \beta - \{r\} \cup \{s\}$, where $r = b_h$. Note that

$$T^\gamma = [\,-q^\gamma \quad A^\gamma\,] = P^{\gamma\beta}\,[\,-q^\beta \quad A^\beta\,]\,, \tag{3.8}$$

where $P^{\gamma\beta} = (B^\gamma)^{-1}B^\beta$ is referred to as the **pivot matrix**.

At the numerical level, a key observation is that $(P^{\gamma\beta})^{-1} = ((B^\gamma)^{-1}B^\beta)^{-1} = (B^\beta)^{-1}B^\gamma$ is a collection of columns from $A^\beta = (B^\beta)^{-1}A$. We think of the entries in column u of $A^\beta = (B^\beta)^{-1}A$ as specifying the linear combination of the columns of B^β that coincides with column u of A, and with one exception, the columns of B^γ are columns of B^β. So, writing a^β_{jk} for the element in row j, column k of A^β, we have

$$
(P^{\gamma\beta})^{-1} = (B^{\beta})^{-1}B^{\gamma} = \begin{bmatrix} 1 & & & a^{\beta}_{1s} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & a^{\beta}_{h-1,s} & & & \\ & & & a^{\beta}_{hs} & & & \\ & & & a^{\beta}_{h+1,s} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & a^{\beta}_{ms} & & & 1 \end{bmatrix} \tag{3.9}
$$

(Although we have represented the matrix as a diagonal matrix with a column replaced, in general it will not be of this form, since the indices of the rows may not be in the same order as the corresponding columns. The essential point is that the column whose h-component is 1 is replaced with the s-column of A^{β}.) Evidently we must have $a^{\beta}_{hs} \neq 0$, else B^{γ} is singular, and by simply multiplying $(P^{\beta\gamma})^{-1}$ by the following matrix to obtain the identity matrix, one can verify that

$$
P^{\gamma\beta} = \begin{bmatrix} 1 & & & -\frac{a^{\beta}_{1s}}{a^{\beta}_{hs}} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & -\frac{a^{\beta}_{h-1,s}}{a^{\beta}_{hs}} & & & \\ & & & \frac{1}{a^{\beta}_{hs}} & & & \\ & & & -\frac{a^{\beta}_{h+1,s}}{a^{\beta}_{hs}} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -\frac{a^{\beta}_{ms}}{a^{\beta}_{hs}} & & & 1 \end{bmatrix} \tag{3.10}
$$

Substituting into (3.8) yields, for any $1 \leq k \leq 2m$

$$
a^{\gamma}_{jk} = \begin{cases} \frac{a^{\beta}_{hk}}{a^{\beta}_{hs}} & \text{if } j = h, \\ a^{\beta}_{jk} - \frac{a^{\beta}_{js}}{a^{\beta}_{hs}} a^{\beta}_{hk} & \text{if } j \neq h. \end{cases} \tag{3.11}
$$

We use the notational convention that $a^{\gamma}_{j0} = -q^{\gamma}_{j}$, and similarly for β. Then analogous formuli for the q^{γ}_{j} are obtained from the above.

The Lemke-Howson algorithm proceeds by "pivoting" along a sequence of i-almost complementary feasible bases, for some given i, at each step maintaining the "tableaux" $T^{\beta} = [\, -q^{\beta} \quad A^{\beta} \,]$ in memory, along with the list of current basis elements and the correspondence between the basis elements and the rows of the tableaux. The specific numerical computations in moving from one basis to the next are given above, and the further description of the algorithm is a matter of specifying the choice of basis to move to next. In general we will want the new basis to be feasible, i-almost complementary, and different from the basis we were at just before arriving at the current basis. If we begin at a complementary feasible solution, then the basis reached on the first pivot will be obtained by adding i or $m+i$ to the basis. If we are at an i-almost complementary feasible basis β that is not complementary, then the new element of the basis will be either j or $m+j$, where j is the omitted index of β. Specifically, the new element will be j $(m+j)$ if $m+j$ (j) was the element that was dropped in the last pivot. If the element that is dropped in this pivot is either i or $m+i$, then the new basis is complementary, and otherwise the new basis is also i-almost complementary.

Geometrically, when we add an element to the basis we are looking at the edge of the set of feasible solutions determined by the equation $Az = q$ and the conditions $z_j = 0$ for all j, other than the one being added, that are not in the basis β. The basic feasible solution corresponding to β is one endpoint of this edge, and the basic feasible solution corresponding to the new basis is the other endpoint. The pivoting procedure described above is the numerical embodiment of this procedure.

For problems whose parameters are generic in the relevant sense, there will always be a unique basis determining the other endpoint of the edge under consideration. It is this case that was worked out by Lemke and Howson. In the general case it can happen that several components of z vanish simultaneously when one reaches the new endpoint, so that any one of these variables could be the one dropped. The procedure described below for resolving this ambiguity is due to Eaves [1971].

A matrix of real numbers is **lex negative** (positive) if the first nonzero entry of each row is negative (positive). A basis is **lex-feasible** if T^β is lex-negative. Note that as long as $q > 0$, the extraneous solution is lex-feasible. Suppose that we are given $T^\beta = [\,-q^\beta \quad A^\beta\,]$, that β is a lex-feasible basis, and we have decided to add s to the basis, so that we must choose an index $r = b_h$ to drop. The choice will be dictated by the requirement that the new basis $\gamma = \beta - \{r\} \cup \{s\}$ be lex-feasible, as we now explain.

We have already seen that we must choose $r = b_h$ with $a^\beta_{hs} \neq 0$. If $r = b_h$ were a feasible choice with $a^\beta_{hs} < 0$, it would have to be the case that $q^\beta_h = 0$, in which case the nonzero components of q^γ are the same as the nonzero components of q^β. In effect the passage from the basis β to the basis γ has not changed the underlying solution. We wish not to allow this possibility, and we therefore require that h should be an element of $S = \{\, j : a^\beta_{js} > 0\,\}$. This set must be nonempty since otherwise the set of feasible solutions would be unbounded. (Any solution would remain a solution if one increased the component of z corresponding to s by 1 while increasing the component corresponding to each h by $-a^\beta_{hs}$.) Let $S_0 = \operatorname{argmin}\{q^\beta_j / a^\beta_{js} : j \in S\,\}$. If the components of q^γ are to be nonnegative, h must be chosen from S_0, clearly. Conversely, if S_0 is a singleton, there is nothing more to it: since $q^\beta_h \geq 0$ and $a^\beta_{hs} > 0$, if $a^\beta_{js} \leq 0$ then $q^\gamma_j \geq 0$ automatically, and otherwise it is nonnegative by virtue of the choice of h. For generic choices of U and q it will be the case that S_0 is always a singleton; such problems are said to be **nondegenerate**. (Eaves [1971] points out that a weaker notion of nondegeneracy suffices.)

When S_0 has more than one element, we must refine our method of choosing which element of the basis to drop. As above, we write $a^\beta_{j0} = -q^\beta_j$, and similarly for γ. Also write $S = S_{-1}$. For $k = 0, \ldots, 2m$ define

$$S_k = \text{argmax}\{ \frac{a^\beta_{jk}}{a^\beta_{js}} : j \in S_{k-1} \}. \tag{3.12}$$

For sufficiently large ℓ, S_ℓ is a singleton (otherwise two rows would be related by scalar multiplication so that the rank of A^β would be less than n), and we let h be its unique member.

Theorem 2 *If β is a lex-feasible basis and $\gamma = \beta - \{r\} \cup \{s\}$ is the basis obtained when s is added to β, as per the procedure described above, then γ is lex-feasible, and β is the lex-feasible basis obtained when r is added to γ, as per the procedure described above.*

Proof: We first show that γ is lex-feasible. We must show that each row of T^γ is lex-negative. Consider row j. If $j = h$, then the row is simply a positive scalar multiple of the corresponding row of T^β, which is lex-negative since β is lex-feasible. If $j \notin S$, so that $a^\beta_{js} \leq 0$, then row j of T^γ is equal to the corresponding row of T^β plus a non-negative scalar multiple of row h of T^β. Since each of these is lex-negative, the resulting sum must be lex-negative. The remaining rows are those with $j \in S_{\ell-1} - S_\ell$, for some $\ell \geq 0$.

It follows that for $k < \ell$, $j \in S_k$, which implies

$$a^\gamma_{jk} = a^\beta_{jk} - a^\beta_{js}\frac{a^\beta_{hk}}{a^\beta_{hs}} = a^\beta_{jk} - a^\beta_{js}\frac{a^\beta_{jk}}{a^\beta_{js}} = 0.$$

For $k = \ell$, the second equality becomes an inequality, yielding $a^\gamma_{j\ell} < 0$. Hence row j is lex-negative. We have established that T^γ is lex-negative, hence γ is lex-feasible.

In the remainder of the proof, we show that β is obtained when r is added to γ. It suffices to show when starting at the basis γ and adding r, that $h \in S_k$ for all k. We will use the fact that, since $r \in \beta$, a^β_{jr} is either 1 or 0 according to whether $j = h$.

Since $a^\beta_{hs} > 0$ was a requirement of the construction of γ,

$$a^\gamma_{hr} = \frac{a^\beta_{hr}}{a^\beta_{hs}} = \frac{1}{a^\beta_{hs}} > 0,$$

it follows that $h \in S = S_{-1}$. Now, for $\ell \geq 0$, assume by way of induction that $h \in S_{\ell-1}$. Then for any $k < \ell$,

$$\frac{a^\gamma_{hk}}{a^\gamma_{hr}} = a^\beta_{hs}\frac{a^\beta_{hk}}{a^\beta_{hs}} = a^\beta_{hk}.$$

For any other $j \in S_{\ell-1}$,

$$a^\gamma_{jr} = a^\beta_{jr} - \frac{a^\beta_{js}}{a^\beta_{hs}}a^\beta_{hr} = -\frac{a^\beta_{js}}{a^\beta_{hs}},$$

it follows that

$$\frac{a^\gamma_{jk}}{a^\gamma_{jr}} = \frac{1}{a^\gamma_{jr}}(a^\beta_{jk} - \frac{a^\beta_{js}}{a^\beta_{hs}}a^\beta_{hk}) = a^\beta_{hk} + \frac{1}{a^\gamma_{jr}}a^\beta_{jk}.$$

Hence, since $a^\gamma_{jr} > 0$

$$\frac{a^\gamma_{hk}}{a^\gamma_{hr}} \geq \frac{a^\gamma_{jk}}{a^\gamma_{jr}} \quad \Leftrightarrow \quad a^\beta_{jk} \leq 0.$$

with equality on the left if and only if there is equality on the right. Thus, if $j \in S_{\ell-1}$, we must have $a^\beta_{jk} = 0$ for all $k < m$. By lex-feasibility of β, it must be that $a^\beta_{j\ell} \leq 0$, which implies that

$$\frac{a^\gamma_{h\ell}}{a^\gamma_{hr}} \geq \frac{a^\gamma_{j\ell}}{a^\gamma_{jr}}.$$

Hence, $h \in S_\ell$. The result now follows by induction on ℓ. ∎

Let $\mathcal{B}^*$ be the set of all complementary lex-feasible bases, and $\mathcal{B}^*_i$ be the set of all i-almost complementary lex-feasible bases. For any $\beta, \gamma \in \mathcal{B}^*_i$, we say that β is **adjacent** to γ if γ is obtained from β by adding either j or $m + j$, as described above, where j is the omitted index for β. If $\beta \in \mathcal{B}^*$, we also say that β is adjacent to $\gamma \in \mathcal{B}^*_i$ if γ is adjacent to β.

The important consequence of the Theorem is that if β is adjacent to γ, then γ is adjacent to β. Any $\beta \in \mathcal{B}^*$ is adjacent to exactly one element in $\mathcal{B}^*_i$ namely the one obtained by adding either i or $m+i$ as the case may be, and any $\gamma \in \mathcal{B}^*_i - \mathcal{B}^*$ is adjacent to precisely two elements in $\mathcal{B}^*_i$, since any adjacent basis must be obtained by adding either j or $m + j$, where j is the omitted index.

Now for any $1 \leq i \leq m$, let $\simeq_i$ be the transitive closure of the adjacency relation on $\mathcal{B}^*_i$. Since $\mathcal{B}^*_i$ is finite, $\simeq_i$ partitions $\mathcal{B}^*_i$ into a finite number of equivalence classes, each of which has a finite number of members. Every member β of an equivalence class is adjacent to either one or two other members. An element that is adjacent to exactly one other member is called an **endpoint**. It follows that each equivalence class must be of the form of a **loop** or a **path**. In a loop, there are no endpoints and every member of the equivalence class is adjacent to exactly two other members. In a path, there are exactly two endpoints, which are connected by a path through the remaining elements. A member β is an endpoint if and only if it is a complementary lex-feasible basis. It follows that $\mathcal{B}^*$ contains an even number of elements. We know that $\mathcal{B}^* \neq \emptyset$, since the extraneous solution is in $\mathcal{B}^*$. This establishes the existence of at least one complementary feasible basic solution other than the extraneous solution. It follows from the above argument that there exists at least one Nash equilibrium. In the non-degenerate case, lex-feasibility is equivalent to feasibility, and the above argument establishes that the number of Nash equilibria is odd.

This leads to the Lemke-Howson algorithm, which amounts to just following the adjacency chain, starting at a complementary lex-feasible basic solution:

1. Pick $\beta_0 \in \mathcal{B}^*$, and $1 \le i \le m$. Find the unique new basis β_1 resulting from adding whichever of i or $m+i$ is not in β_0, as per the procedure above. Set $k = 1$.

2. If $\beta_k \in \mathcal{B}$, halt. Otherwise proceed to 3.

3. Given β_k from which j (resp. $m + j$) has just been dropped, add $m + j$ (resp. j) and find the unique new basis β_{k+1} allowed by the procedure above. Set $k = k + 1$ and return to 2.

From the above remarks, we see that the algorithm cannot cycle. Since there are finitely many bases, the algorithm must eventually halt. Shapley [1974] points out that in the non-degenerate case, the Lemke-Howson algorithm connects equilibria with opposite indices. Since the extraneous solution has index of -1, this means that any equilibria reached by starting from the extraneous solution must have index of $+1$.

We have described the Lemke Howson algorithm for a somewhat larger class of problems than that generated by two person games. In the case of a two person game, U has a partitioned form, as in equation (3.3). In this case, by rearranging the columns, we can write

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, A_i = [\, I_i \quad U_{-i} \,] .$$

Then, we can write B^β in the form

$$B^\beta = \begin{bmatrix} B_1^\beta & 0 \\ 0 & B_2^\beta \end{bmatrix},$$

It follows that

$$T^\gamma = P^{\gamma\beta} T^\beta = \begin{bmatrix} -P_1^{\gamma\beta} q_1^\beta & P_1^{\gamma\beta} A_1^\beta & 0 \\ -P_2^{\gamma\beta} q_2^\beta & 0 & P_2^{\gamma\beta} A_2^\beta \end{bmatrix},$$

where $P_i^{\gamma\beta} = (B_i^\gamma)^{-1} B_i^\beta$, $q_i^\beta = (B_i^\beta)^{-1}\mathbf{1}$, and $A_i^\beta = (B_i^\beta)^{-1} A_i$. Thus, the pivot operation leaves the zero blocks of A untouched. For the same reason, the above shuffling of the columns does not affect lex-feasibility. Further, when β and γ differ by only one element, one of $P_i^{\gamma\beta}$, $i = 1, 2$ must be an identity matrix.

From a numerical point of view the above observations mean that for two person games, we need not do computations on (or even store) the whole tableau. Rather, the tableau can be decomposed into two smaller tableaus $T_i^\beta = [\, -q_i^\beta \quad A_i^\beta \,]$, which can be computed on independently, since any pivot operation only affects one of them. (In fact the Lemke-Howson algorithm requires that pivot operations alternate between the two tableaus.) Further computational simplification results since one only has to maintain data for the current non-basic columns of each tableau.

3.1.2 Finding multiple Nash equilibria

In order to initiate the Lemke-Howson algorithm, one needs to choose an index $1 \le i \le m$, and a complementary lex-feasible basic solution (*clbs*) $\beta_0 \in \mathcal{B}^*$. Typically, one will start with β_0 being the extraneous solution, but this is not necessary. The algorithm can be initiated from any $\beta \in \mathcal{B}^*$, as long as one can find it. This suggests a means of using the Lemke-Howson algorithm to find more than one solution. For any set $\mathcal{B}' \subseteq \mathcal{B}^*$ define

$$\mathcal{A}_\ell(\mathcal{B}') = \{\beta \in \mathcal{B}^* : \beta \simeq_\ell \beta' \text{ for some } \beta' \in \mathcal{B}'\},$$

and

$$\mathcal{A}(\mathcal{B}') = \cup_{\ell=1}^m \mathcal{A}_\ell(\mathcal{B}').$$

Thus $\mathcal{A}(\mathcal{B}')$ is the set of *clbs*'s that are **immediately accessible** from $\mathcal{B}'$ via the Lemke-Howson algorithm. For any integer t, define $\mathcal{A}^t(\mathcal{B}') = \mathcal{A}(\mathcal{A}^{t-1}(\mathcal{B}'))$, where we define $\mathcal{A}^o(\mathcal{B}') = \mathcal{B}'$. Since $\mathcal{B}^*$ is finite, clearly, there must exist a T for which $\mathcal{A}^t(\mathcal{B}') = \mathcal{A}^r(\mathcal{B}')$ for all $r, t \geq T$. Define $\mathcal{A}^*(\mathcal{B}') = \mathcal{A}^T(\mathcal{B}')$. This is the set of *clbs*'s that are **accessible** via the Lemke Howson algorithm from $\mathcal{B}'$.

If we want to find all Nash equilibria to a normal form game, one might try to set $\mathcal{B}' = \{\beta_0\}$, where β_0 is the extraneous solution, and then use successive applications of the Lemke-Howson algorithm to find $\mathcal{A}^*(\mathcal{B}')$. One would then hope that $\mathcal{A}^*(\mathcal{B}') = \mathcal{B}^*$. Unfortunately, Wilson constructed an example of a two person game (reported in Shapley [1974]) for which $\mathcal{A}^*(\mathcal{B}') \neq \mathcal{B}^*$. The example is generic, in the sense that small perturbations of the payoffs do not change the property that $\mathcal{A}^*(\mathcal{B}') \neq \mathcal{B}^*$. Thus, we cannot in general hope to find all equilibria by computing the accessible equilibria. Shapley [1981] has shown that it is possible to perturb a game so that the set of Nash equilibria remains unchanged, but the accessibility relation changes. However, he does not provide any general method of perturbing a game so that all Nash equilibria become accessible.

3.1.3 Zero sum games

For two person zero sum games, the problem of finding Nash equilibria simplifies considerably, as we are able to express the problem as a linear program.

The value of a game is defined as the maximum that a player can guarantee him/-herself irrespective of the strategy of the opponent. For a two person game, the value of the game, v_i, to player i is expressed as the solution to the following linear program:

$$\begin{array}{lrl} \text{minimize} & v_i & \\ \text{subject to} & U_i \cdot p_{-i} & \leq v_i \cdot \mathbf{1} \\ & p_i & \geq 0 \end{array} \tag{3.13}$$

It follows from the minimax theorem that for two person zero sum games, $v_1 = -v_2$, and that a strategy pair is minimax if and only if it is a Nash equilibrium. Hence, the set of Nash equilibria is exactly the set of solutions to the above linear program. Since the set of solutions to a linear program is a convex polyhedron, it follows that the set of all Nash equilibria is a convex set, which can be completely characterized by a finite number of extreme points. These are just the set of optimal basic feasible solutions of the above linear program.

3.1.4 Summary

In summary, the Lemke-Howson algorithm provides a way to find at least one Nash equilibrium for any two person game. As long as the data of the problem are rational, the algorithm can provide exact solutions, since all computations are done in the rational field. The Lemke-Howson algorithm is guaranteed to eventually find a solution in any problem. But, because there is no objective function in the Lemke-Howson algorithm, there is no way of determining how close one is to a solution during the computation.

Regarding the computational complexity of the Lemke-Howson algorithm, Murty [1978] showed an exponential lower bound for Lemke's algorithm when applied to linear complementarity problems. However, there are no results giving the computational complexity of the Lemke-Howson algorithm for finite two person games.

The Lemke-Howson algorithm can also sometimes be used to find multiple Nash equilibria. It can compute a set of *accessible* Nash equilibria. But there is no guarantee that the accessible equilibria include all Nash equilibria.

In two person, zero sum games, the problem reduces to a linear program. In this case, the set of *all* Nash equilibria is a convex set, which can be characterized as the set of optimal basic feasible solutions to the same linear program.

3.2 N-Person Games: Simplicial Subdivision

For n-person games, with n greater than two, the problem of finding a Nash equilibrium is no longer a linear complementarity problem, so we can not apply the Lemke-Howson algorithm any longer. Rosenmüller [1971] and Wilson [1971] independently extended the Lemke-Howson algorithm to find Nash equilibria for n-person games, but this extension requires following a solution set of a set of non linear equations, and has not (to the authors' knowledge) been implemented in computer code. More tractable approaches use path following approaches derived from Scarf's algorithm for finding fixed points for a continuous function on a compact set.

3.2.1 Fixed triangulation

The simplicial subdivision algorithms derive from the work of Scarf [1967, 1973]. Most of the original work in this area deals with finding fixed points of a function $f : \Delta^n \mapsto \Delta^n$ defined on an $(n-1)$-dimensional simplex, $\Delta^n = \{p = (p_1, \ldots, p_n) : \sum_{i=1}^n p_i = 1, p_i \geq 0\}$. For application to game theoretic problems, we need a version which operates on the product of unit simplices, $\Delta = \prod_i \Delta_i$, called the **simplotope**. We describe a version of a simplicial subdivision algorithm that works on this product set.

It is convenient to represent player i's j^{th} pure strategy as $s_{ij} = e_{ij}$, where e_{ij} is the i, j^{th} basis vector in $\mathbb{R}^m$. Thus e_{ij} is the vector with a 1 in element $\sum_{l=1}^{i-1} m_l + j$, and 0 everywhere else, and player i's set of pure strategies is $S_i = \{e_{i1}, \ldots, e_{im_i}\}$. We can write $\Delta_i = \sigma(S_i) = \sigma(e_{i1}, \ldots, e_{im_i})$, and $\Delta = \prod_i \Delta_i = \prod_i \sigma(S_i)$.

The algorithm treats each agent's set of pure strategies as an ordered set. There are no constraints on these orderings, and we will use the ordering given by the indexation. Thus a set $T \subseteq \cup_i S_i$ is said to be **ordered** if, for all i, there is a $k_i \geq 0$ such that $T_i = \{s_{ij} : j \leq k_i\}$. For such an ordered set define $\overline{T}_i = T_i \cup \{s_{i,k_i+1}\}$.

We now give a formal definition of simplicial subdivision. For any $t+1$ affinely independent points $\{v_0, \ldots, v_t\}$ in $\mathbb{R}^r$, the convex hull $\sigma = \sigma(v_0, \ldots, v_t)$ of $\{v_0, \ldots, v_t\}$ is called a t dimensional **simplex** with vertices $v_0, \ldots, v_t$. A simplex τ is a **face** of σ if the set of vertices of τ is a subset of the vertices in σ. A **proper face** is a face that is not all of σ; the **interior** of σ is the set of points in σ that are not elements of some proper face. If τ is a face of σ with one less vertex than σ, then τ is called a **facet** of σ. A finite collection G of simplices is a **triangulation** of a compact, convex set C if the interiors of the faces of all simplices partition C. An equivalent condition is that the union of the simplices in G is C, and if two simplices have a nonempty intersection, their intersection is a face of each.

Suppose that we are given a triangulation G of the simplotope Δ. Then G induces a triangulation of each face of Δ, as we now explain. For any $T \subseteq \cup_i S_i$, define $T_i = T \cap S_i$. Any face of Δ is of the form $\prod_i \sigma(T_i)$ where each $T_i \subseteq S_i$ is nonempty. If the interior of a face of a simplex in G intersects $\prod_i \sigma(T_i)$, then that face is entirely contained in $\prod_i \sigma(T_i)$. Every point in $\prod_i \sigma(T_i)$ is in the interior of some simplex in G, and of course the interiors of such simplices are pairwise disjoint, since this is true for G.

A **labeling** on Δ is a function $l : \Delta \mapsto \cup_i S_i$. Assume that T is an ordered set, and let $\sigma \in G$ be a simplex in $\prod_i \sigma(\overline{T}_i)$, of maximal dimension, so that σ has $1 + \sum_i \#(T_i)$ vertices. Then σ is said to be **almost completely labeled** if the labels associated with the vertices of σ include each T_i. An almost completely labelled simplex σ is **completely labeled** if the labels associated with the vertices of σ include $\overline{T}_i$ for some $i \in N$, in which case σ is called an ***i*-stopping simplex**. The goal of the algorithm will be to find such a simplex for any labelling that is **Sperner-proper**: for all $T \subseteq \cup_i S_i$ with $T_i \neq \emptyset$ for all i, $l(\prod_i \sigma(T_i)) \subseteq T$. Concretely, a labeling is Sperner proper if, for any $v \in \Delta$, $v_{ij} = 0 \Rightarrow l(v) \neq e_{ij}$.

Let Σ_T denote the set of almost completely labeled simplices of the ordered set $T \subseteq S$. Let $\Sigma = \cup_T \Sigma_T$, where the union is over all ordered sets T. We now define **adjacency**, a binary relation on Σ.

Unless $\sigma \in \Sigma_T$ is 0-dimensional, which happens precisely when $T_i = \emptyset$ for all i, any $\sigma \in \Sigma_T$ has at least one facet τ with an almost complete set of labels– i. e., $l(\tau) = T$. Possibly τ is also a facet of one other $(\sum_i \#(T_i))$-dimensional simplex in $\prod_i \sigma(T_i)$, say σ'. Since $\tau \subset \sigma'$, σ' is completely labelled. In this case σ and σ' are adjacent. If τ is not a face of two simplices of maximal dimension in $\prod_i \sigma(\overline{T}_i)$, then τ must be contained in the boundary of this set, so that $\tau \subset \sigma(\overline{T}_j - \{s_{jl}\}) \times \prod_{i \neq j} \sigma(\overline{T}_i)$ for some j and $s_{jl} \in S_j$. Since l is Sperner proper, s_{jl} cannot be a label of a vertex of τ, i.e., $s_{jl} \notin T$, so we must have $s_{jl} = s_{jk_j}$. Two ordered sets T and T' are said to be **adjacent** if they differ by at most one element, say $s_{jk_j} \in T - T'$. When T and T' are adjacent, a simplex $\sigma \in \Sigma_T$ and a simplex $\tau \in \Sigma_{T'}$ are adjacent if $l(\tau) = T$ (so τ is completely labeled) and τ is a facet of σ. The definition of adjacency is now complete, in the sense that there are no cases in which two simplices in Σ are adjacent aside from the two just described.

Summarizing, given $\sigma \in \Sigma_T$ and a facet τ with an almost complete set of labels, σ is adjacent either to τ itself or to another simplex in Σ_T which also has τ as a facet. Let v be the vertex in $\sigma - \tau$. If $l(v) \in T$, then σ has one other facet with an almost complete set of labels, so that σ must be adjacent to precisely two other simplices. If $l(v) \notin T$, then σ is completely labeled, say with $l(v) = s_{j,k_j+1}$, then σ is adjacent to the unique almost completely labeled simplex in $\sigma(\overline{T}_j \cup \{s_{jk_j+2}\}) \times \prod_{i \neq j} \sigma(\overline{T}_i)$, *unless* $k_j + 1 = m_j$, *so that the labels for* σ *include all the pure strategies of* j.

By the definition of adjacency, every almost completely labeled simplex $\sigma \in \Sigma_T$ for some ordered set $T \subseteq S$ is adjacent to at most two other simplices. A simplex that is adjacent to at most one other simplex is called a **terminal simplex**. It follows that if $\sigma \in \Sigma$ and J is the set of simplices in Σ that can be reached by starting at σ and proceeding along a sequence of simplices in which each simplex is adjacent to its predecessor, then J must be: (a) a **loop** – there is no terminal simplex; (b) a **string** – there are two terminal simplices; or (c) a **point** – J contains a single (terminal) simplex adjacent to no other simplex. A terminal simplex must either be the 0-dimensional simplex σ_0 that is the unique element of $\prod_i \sigma(\overline{T}_i)$ when $T_i = \emptyset$, an i-stopping simplex for some $i \in N$, or both. The last possibility (namely a point) is ruled out if we require that $m_i \geq 2$ for all i. With this trivial case eliminated, it follows that there are an odd number of i-stopping simplexes.

Given algorithmic procedures for elaborating the triangulation and computing labels, the above results define an algorithm for finding a stopping simplex: start at σ_0 and follow the adjacency relation to the other endpoint of the string. In order for a stopping simplex to be of interest, it must approximate a Nash equilibrium to the game. We now present a labeling function which achieves this.

For any n-person game, and any $p \in \Delta$, we define, as in section 2, equation (2.5)

$$g_{ij}(p) = max[u_i(s_{ij}, p_{-i}) - u_i(p), 0] \quad \text{and} \quad y_{ij}(p) = \frac{p_{ij} + g_{ij}(p)}{1 + \sum_j g_{ij}(p)}.$$

As noted previously, $y : \Delta \mapsto \Delta$ is a continuous function whose fixed points are precisely the Nash equilibria. For $p \in \Delta$, define $l(p) = s_{ij}$, where (i,j) is the lexicographic least index in $\text{argmin}_{l \in N, 1 \leq k \leq m_l} y_{lk}(p) - p_{lk}$.

Now let $\{G^r\}_{r=1}^\infty$ be a sequence of triangulations whose meshes converge to 0. For each r let σ^r be an i-stopping simplex for some i. Since we may pass to a subsequence, we may assume that it is the same i for all r, and we may also assume that the sequence σ^r converges, in the obvious sense, say to p^*. Now for each $s_{ij} \in S_i$ and each r there is a vertex v of σ^r with $l(v) = s_{ij}$. Among other things this implies that $y_{ij}(v) \leq v_{ij}$, and passing to the limit yields $y_{ij}(p^*) \leq p^*_{ij}$. This is true for all $1 \leq j \leq m_i$, so we must have $y_{ij}(p^*) = p^*_{ij}$ for all $1 \leq j \leq m_i$. In view of the definition of l, it follows that $0 = \min_{l \in N, 1 \leq k \leq m_l} y_{lk}(p) - p_{lk}$, so for all $k \in N$ and $1 \leq h \leq m_k$, $y_{kh}(p^*) \geq p^*_{kh}$. But then $y_{kh}(p^*) = p^*_{kh}$ for each k and all relevant h. That is, p^* is a fixed point of y, hence a Nash equilibrium.

Thus we have described an algorithm which, for given ϵ, halts at a point $p \in \Delta$ with $\|y(p) - p\| < \epsilon$, namely execute the procedure for finding an i-stopping simplex on a sequence of successively finer triangulations of Δ until some vertex of such a simplex satisfies the halting criterion. It should be stressed that we have *not* given a procedure for finding a point that is within ϵ of some fixed point of y. This is inherent in any algorithm that assumes only that y is continuous, and which uses only the values of y at particular points to regulate its behavior. To see this imagine such an algorithm halting after computing the value of y at some sequence of points $p_1, \ldots, p_z$. Then the algorithm would behave in the same way if y was replaced by $h^{-1} \circ f \circ h$ where $h : \Delta \to \Delta$ is a homeomorphism that leaves $p_1, \ldots, p_z$ and $y(p_1), \ldots, y(p_z)$ fixed, but by choosing h appropriately we are free to locate the fixed points of $h^{-1} \circ f \circ h$ almost arbitrarily.

In fact for the problem of finding Nash equilibrium, additional information is available, in that the given functions are algebraic. One could imagine combining the procedure above with an additional step using this information to test whether the candidate approximate equilibrium was, in fact, near an actual equilibrium. To our knowledge this issue has not been investigated.

3.2.2 Refining the triangulation

One of the problems with the Scarf type simplicial subdivision algorithms described above is that they depend on a given triangulation. If one finds an approximation to a solution using a triangulation G with a given mesh, and one then wants to get a better approximation, one must start all over with a new triangulation with a smaller mesh. The computational effort that went into finding an original approximation does not help in finding a finer approximation.

This problem has been dealt with in two ways. The first is the **homotopy method**, originally developed by Eaves [1972]. Here the idea is to add a dimension to the problem representing the accuracy of the approximation, say $t \in [0, 1]$. Then one triangulates the product space $\Delta \times [0, 1]$ in such a way that (roughly) as t approaches 1, the mesh of the triangulation approaches 0. Doup and Talman [1987b] provide a triangulation which is suitable for application of this method to the simplotope, and which allows for arbitrary rate of grid refinement.

The second approach to getting better approximations is the **restart method**. Here the idea is to develop path following algorithms that allow for start at an arbitrary point in the solution space. Merrill [1972], Kuhn and MacKinnon [1975], and van der Laan and Talman [1979] developed methods that allow restart at an arbitrary point in the space, with a triangulation of smaller mesh. In a series of articles, Van der Laan and Talman [1979, 1980, 1982], van der Laan, Talman and van der Heyden [1987] and Doup and Talman [1987a] have developed versions of these algorithms which can be applied to the simplotope, and also investigated in considerable detail the advantages of different triangulations.

3.2.3 Computational complexity

Hirsch, Papadimitriou and Vavasis [1989] showed that Scarf's algorithm, (actually *any* algorithm for computing a Brouwer fixed point based on evaluation of function values), has a worst case complexity which is exponential in the dimension and the number of digits of accuracy. Todd [1982] showed an exponential lower bound for piecewise-linear path-following algorithms applied to triangulations of affine linear problems.

On the other hand Saigal [1977] shows that if the underlying function is continuously differentiable and satisfies a Lipschitz condition, that simplicial subdivision methods can achieve quadratic rates of convergence (in the limit) if the change in mesh size is chosen appropriately.

One problem with restart methods for use on game theory problems that apparently does not arise in fixed point problems on the simplex is related to the definition of a stopping simplex when one works in the simplotope rather than in a single simplex. The results discussed above guarantee us that as we refine the triangulations, sending the mesh size to zero, there will be a sequence of stopping simplices converging to a Nash equilibrium. However, we are not guaranteed that the sequence of stopping simplices will be stopping simplices for the same player. After a restart, the search for the right player with a stopping simplex can sometimes cause the algorithm to go out to the boundary of the simplotope before returning to the approximate starting location. It is possible that these problems may be avoided in the homotopy methods applied to the simplotope, or in algorithms such as that of Garcia, Lemke and Luethi [1973] who, instead of using the simplotope, embed the problem in a simplex of dimension $m - 1$. Unfortunately, there is not much in the way of comparative computational experience of these algorithms as applied to game theory problems reported in the literature.

3.3 Non Globally Convergent Methods

In this section, several methods for finding Nash equilibria are presented which are not globally convergent. Despite the fact that they do not satisfy global convergence, these methods are important because they offer other features, such as speed or the ability to find a particular equilibrium for the purposes of doing comparative statics, for example.

3.3.1 Nash equilibrium as non-linear complementarity problem

We saw in section 2 that an n-person game can be represented as a non-linear complementarity problem. Mathiesen [1987] and others have been very successful at finding economic equilibria by formulating the problem as a non-linear complementarity problem, and then solving that problem by a sequence of approximations by linear complementary problems (SLCP). Harker and Pang [1990] survey recent developments in this area as well as the more encompassing set of variational inequality problems.

The general idea of the SLCP approach is that one picks a starting point $p^0 \in \Delta$ in the simplex, and then approximates the non-linear problem at that point with the linear complementarity problem obtained by taking the first order Taylor expansion around the point. One then uses Lemke's algorithm for the linear complementarity problem to find an exact solution to that problem. This gives a new point, $p^1 \in \Delta$, from which one can repeat the process. One then hopes the sequence $\{p^t\}$ converges to a solution of the original non-linear complementarity problem. The method can be thought of as a generalization of Newton's algorithm for finding local optima of a C^2 function, and is also related to the "global Newton method" of Smale [1976].[1] Like Newton's method, the SLCP method is not globally convergent, and requires judicious choice of a starting point.

Since an n-person game can be formulated as a non-linear complementarity problem also, the SLCP method should be applicable to solving large normal form games. Van den Elzen and Talman [1992] have used similar ideas in the computation of game theoretic equilibria. They formulate the Nash equilibrium as a stationary point problem, and then approximate the stationary point problem by a sequence of linear stationary point problems.

These methods do not satisfy global convergence. However, they do have the advantage of speed. Hence even without global convergence, the ability to repeatedly try a number of starting points if convergence fails can make these methods attractive, especially in large problems, if the only goal is to find a sample Nash equilibrium.

3.3.2 Nash equilibrium as a minimum of a function

As discussed in section 2, the problem of finding a Nash equilibrium can be reformulated as a problem of finding the minimum of a real valued function on a polytope, where the global minima of the function correspond to the Nash equilibria of the underlying problem. Under this approach, every isolated Nash equilibrium has a basin of attraction. So if one starts close enough to an isolated Nash equilibrium, then one can guarantee to find it with any level of accuracy desired.

Recall, we defined the function $v : \mathcal{P} \mapsto \mathbb{R}$ in equation (2.9) by

$$v(p) = \sum_{i \in N} \sum_{1 \leq j \leq m_i} [g_{ij}(p)]^2.$$

First, note that $v(p)$ is a non-negative function that is zero if and only if p is a Nash equilibrium for the game. Further, as is shown in McKelvey [1992], v is everywhere differentiable.

We want to minimize $v(p)$ subject to the constraints that $\sum_j p_{ij} = 1$ and $p_{ij} \geq 0$ for all i, j. One can use methods for constrained optimization to find solutions. Alternatively, we can impose the constraints as penalty functions, yielding a revised version of the objective function:

[1] Contrary to the suggestion of the name, the global Newton is not globally convergent. See e.g., Doup [1988].

$$w(p) = v(p) + \sum_{ij} \{min[p_{ij}, 0]\}^2 + \sum_{i \epsilon N} (1 - \sum_j p_{ij})^2.$$

This is also a differentiable function, defined over all of $\mathcal{P}$, and p^* is a Nash equilibrium if and only if it $w(p^*) = 0$. Any of a number of standard algorithms for finding the unconstrained minimum to a function on $\mathbb{R}^m$ can be applied to find the minima of w.

Note that it was not established in the above characterization that all minima to the function v or w are global minima. Hence, there may be local minima to the objective function that are not global minima, and hence not Nash. So it is important to check the value of the objective function after convergence, to verify that the point found is indeed a Nash equilibrium.

The speed of the algorithm (in the authors' experience) is generally slower than other methods, and as is evident from the discussion in the preceding paragraph, the algorithm may sometimes require judicious choice of a starting point. Nevertheless, there are some situations in which the algorithm is preferred to existing methods. The path following algorithms discussed above are not capable of finding all of the Nash equilibria. Even if one uses a restart algorithm to start them close to a given Nash equilibrium, there is no guarantee they will find the nearby equilibrium. However, every isolated Nash equilibrium will have an open region around it where the value of the objective function v is strictly greater than 0. Hence, every Nash equilibrium will have a radius of convergence such that if one starts within this radius of the Nash equilibrium, and uses any descent method which guarantees that the objective function decreases at at each step, then the algorithm proposed in this section will converge to the nearby Nash equilibrium. Therefore, the method proposed here is useful for investigating the comparative statics of a particular equilibrium, as a function of the payoffs of the game.

The function v can also be used to define a differential equation system whose zeros are Nash equilibria (see McKelvey [1992]). However, since there may be local minima that are not global minima, the system is not globally convergent. Other work has been done on differential equation systems for solving for Nash equilibria. Brown and von Neumann [1950] construct a differential equation systems for finite two person zero sum games which is globally convergent. Rosen [1964] defines a differential equation system for n-person games, and he gives conditions for global convergence. Rosen's conditions require a property of "diagonal strict concavity" on the matrix of cross derivatives of the payoff functions, which would not in general be satisfied for finite n-person games.

4 Extensive Form Games

To this point we have discussed only normal form games, ignoring all computational issues that might arise out of the derivation of the normal form from some underlying extensive form game. This is reflective of the state of the literature, which has only recently begun to address computation of extensive form equilibrium concepts. Nevertheless, the possibility of using information in the extensive form to guide computation has enormous intuitive appeal, and is one of the principle factors that motivated Kreps and Wilson (1982) (henceforth KW) to develop the concept of sequential equilibrium, which we now review. With minor modifications, our notation is that of KW.

4.1 Notation

In an extensive form game the set of physically possible sequences of events is described by a tree or, somewhat more generally, an **arborescence** (the accepted term in computer science is 'forest'), which may be thought of as a finite collection of trees. Let $(T, \prec)$ be a pair consisting of a finite set T of **nodes** and a binary relation $\prec$ on T. We interpret $x \prec t$ as meaning that the node x **precedes** the node t, and we require that $\prec$ be transitive and acyclic. In addition we require that, for any $t \in T$, the set $P(t) = \{\, x \in T : x \prec t \,\}$ of predecessors of t is completely ordered by $\prec$. In effect this amounts to a decision to treat equivalent positions reached by different sequences of moves (e.g., the positions reached in chess after 1. P-Q4, N-KB3; 2. P-QB4 and 1. P-QB4, N-KB3; 2. P-Q4) as distinct. For most purposes this is a harmless simplification, but we hasten to point out that it is far from benign from the point of view of minimizing computational burden.

The set of **initial nodes** is $W = \{\, w \in T : P(w) = \emptyset \,\}$, and $Y = T - W$ is the set of **noninitial nodes**. Similarly, the set of **terminal nodes** is $Z = \{\, z \in T : P^{-1}(z) = \emptyset \,\}$, and $X = T - Z$ is the set of **nonterminal** or **strategic nodes**. The assumption that $P(t)$ is always completely ordered implies that the **immediate predecessor** function $p_1 : Y \to X$, $p_1(y) = \max P(y)$, is well defined. We let $\ell(t)$ be the cardinality of $P(t)$, and for $\ell \geq 2$ we define $p_\ell : \{\, y \in Y : \ell(y) \geq \ell \,\} \to X$ to be the ℓ-fold composition of p_1 with itself. Also, let p_0 be the identity function on T. The number of predecessors of $t \in T$ is $\ell(t)$, the largest integer such that $p_{\ell(t)}(t)$ is defined.

We adopt the convention that 0 represents "nature" or "chance." There are finite sets H, A, and $I = \{\, 1, \ldots, n \,\}$ of **information sets**, **actions**, and **agents**, respectively, and functions $\eta : X \to H$, $\alpha : Y \to A$, and $\iota : H \to \{\, 0 \,\} \cup I$, interpreted as follows. The **information partition** is the collection of sets of the form $\eta^{-1}(h)$, $h \in H$. (The information partition is taken as given in most formal definitions of extensive form games, which do not include the function η.) We will often abuse notation by equating h with $\eta^{-1}(h)$ so that we write $x \in h$ to indicate $\eta(x) = h$. For $h \in H$, the agent who chooses an action when a node in h occurs is $\iota(h)$. Let $H^i = \iota^{-1}(i)$ be the set of information sets at which i chooses, $i = 0, \ldots, n$. When an action is chosen at $h \in H^i$, agent i knows that some node in $\eta^{-1}(h)$ has occurred, but not which one. For $y \in Y$, $\alpha(y)$ is interpreted as the action that was chosen at the immediate predecessor of y, so that $A(x) = \alpha({p_1}^{-1}(x))$ is the set of actions that are available at $x \in X$. To make sense out of the choice problems in this structure we must assume that: (a) if $y, y' \in Y$ are distinct with $p_1(y) = p_1(y') = x$, then $\alpha(y) \neq \alpha(y')$; (b) $A(x) = A(x')$ for all h and $x, x' \in h$. For $h \in H$ let $A(h)$ be the set of actions that can be chosen at h. It is conventional to assume that $A(h) \cap A(h') = \emptyset$ for distinct $h, h' \in H$.

In KW it is assumed that nature has no decisions, so the range of ι is I, and all 'physical' uncertainty is summarized in the **initial assessment** $\rho \in \Delta(W)$. For many purposes this is a harmless simplification — one can shuffle once at the beginning of the game, rather than before each card is drawn — but again this may not be computationally efficient, so we do not adopt this assumption. The behavior of nature at the information sets where it chooses is represented by π^0, a vector of probability measures $\pi^0_h \in \Delta(A(h))$. (The information partition of the nodes at which nature chooses will play no essential role in the theory. It will be easy to see that postulating this structure entails no loss of generality, since we could have each node at which nature chooses be a singleton information set.)

In games played by teams it can be sensible to allow an agent (construed as the team) to forget information from one move to the next, but we will take the usual perspective and assume that this does not happen. What this means precisely is that the game is one of **perfect recall**: for any information set h and any $x, x' \in h$, if x^* is a predecessor of x at which the same agent chooses (that is, $\iota(\eta(x^*)) = \iota(\eta(x))$) then $\eta(x^*)$ contains a node x'^* that is a predecessor of x', and if y and y' are the nodes with $y \in \{x\} \cup P(x)$, $p_1(y) = x^*$ and $y' \in \{x'\} \cup P(x')$, $p_1(y') = x'^*$, then $\alpha(y) = \alpha(y')$. More verbally, at $\eta(x)$, $\iota(\eta(x))$ knows that $\eta(x^*)$ occurred, and that at that point $\alpha(y)$ was chosen; we require that this is also true when x' occurs.

A **behavior strategy** for agent i is a vector $\pi^i = (\pi^i_h)_{h \in H^i}$ of probability measures $\pi^i_h \in \Delta(A(h))$. Let $\Pi^i = \prod_{h \in H^i} \Delta(A(h))$ be the set of such objects. The interpretation is that, conditional on h being reached, for $a \in A(h)$, $\pi^i_h(a)$ is the probability that a will be chosen. The term **behavior strategy** is also used to refer to vectors $\pi = (\pi^i)_{i \in I}$ whose components are behavior strategies for the various agents. Let $\Pi = \prod_{i \in I} \Pi^i$ be the space of behavior strategies in this sense. (Formally we do not include Π^0 as a factor of Π, but in many expressions we will treat the given π^0 as a component of π.) It simplifies many expressions to write π_h and π_x in place of $\pi^{\iota(h)}_h$ and $\pi^{\iota(\eta(x))}_{\eta(x)}$ respectively. Given a behavior strategy π, the probability that a node t is reached is

$$\mathbf{P}^\pi(t) = \rho(p_{\ell(t)}(t)) \cdot \prod_{\ell=1}^{\ell(t)} \pi_{p_\ell(t)}(\alpha(p_{\ell-1}(t))),$$

and if $x \prec t$, then the probability that t will be reached conditional of x having been reached is

$$\mathbf{P}^\pi(t|x) = \prod_{\ell=1}^{\ell(t)-\ell(x)} \pi_{p_\ell(t)}(\alpha(p_{\ell-1}(t))).$$

Recently Koller and Megiddo [1992], von Stengel [1993], and Koller, Megiddo, and von Stengel [1994] have pointed out that there are computational advantages in working with the **sequence form**, which is essentially a representation of behavior strategies in terms of the induced probabilities, for each agent and each sequence of actions for that agent, that the sequence will be chosen if nature and the other agents behave in a way that permits the sequence. More precisely, suppose that $s = (a_1, \ldots, a_q)$ is a sequence of actions that might be chosen by agent i, meaning that the information set at which a_1 might be chosen can occur without any earlier choices by agent i, and, for $j = 2, \ldots, q$, the information set at which a_j may be chosen can occur after a_{j-1} is chosen, without any intervening actions by agent i. For each such sequence for agent i set $\beta_i^{\pi_i}(s) = \prod_{j=1}^{q} \pi_i(a_j)$; this defines an induced **realization plan** $\beta_i^{\pi_i}$ for agent i. The critical point is that for each node t, $\mathbf{P}^\pi(t)$ is a product of sequence form variables, with one factor for each agent. As a monomial in sequence form variables, the degree of $\mathbf{P}^\pi(t)$ can easily be much lower than its degree as a monomial in behavior strategy variables. It is computationally trivial to pass from a realization plan to the set of behavior strategies that induce it, so for many solution concepts it is likely to be efficient to solve for equilibria in terms of the sequence form, then pass to behavior strategies. This is not possible for sequential equilibrium, since the sequence form suppresses information about an agent's 'intended' behavior at information sets that the agent never allows to be reached, whereas sequential equilibrium imposes conditions on such intentions. Nonetheless, it seems likely that a suitable generalization of the sequence form will be the natural vehicle for computation of sequential equilibrium.

When every possible event has positive probability, there are unambiguous conditional probabilities defining **beliefs** at the various information sets. For $i \in I$ let $\Pi^{i\circ} = \prod_{h \in H^i} \Delta^\circ(A(h))$ be the set of **totally mixed** behavior strategies for i, and let $\Pi^\circ = \prod_{i \in I} \Pi^{i\circ}$. For a totally mixed behavior strategy π, each node t is reached with positive probability, so for each h there is an induced probability distribution $\mu_h \in \Delta(h)$ given by

$$\mu_h^\pi(x) = \mathbf{P}^\pi(x) / \sum_{x' \in h} \mathbf{P}^\pi(x').$$

The vector $\mu^\pi = (\mu_h^\pi)_{h \in H} \in M \equiv \prod_{h \in H} \Delta(h)$ is called the **belief** induced by π. The equilibrium concept we are working towards requires not just ex ante rationality, but also rationality, conditional on certain beliefs, at information sets that have no probability of being reached under the equilibrium strategies. Thus there arises the question of what is the appropriate requirement concerning the relation between the behavior strategy and the belief. KW define the set of **interior consistent assessments** to be $\Psi^\circ = \{ (\mu^\pi, \pi) : \pi \in \Pi^\circ \}$, and the set Ψ of **consistent assessments** is its closure in $M \times \Pi$. KW express doubts about this notion of consistency, but recently Kohlberg and Reny [1993] have shown that it is equivalent to a conceptually natural condition that is slightly stronger than independence.

We now introduce a **payoff** $u \in \mathbb{R}^{Z \times I}$ which specifies a reward for each agent at each terminal node. A behavior strategy induces an expected payoff

$$E^\pi(u_i|t) = \sum_{z \in Z} \mathbf{P}^\pi(z|t) \cdot u_i(z)$$

for each $i \in I$ at each $t \in T$, and an assessment (μ, π) induces an expected payoff

$$E^{(\mu,\pi)}(u_i|h) = \sum_{x \in h} \mu_h(x) E^\pi(u_i|x)$$

for each $i \in I$ at each $h \in H^i$. The assessment (μ, π) is **sequentially rational** if, for each $i \in I$ and $h \in H^i$, there does not exist $\hat{\pi}^i \in \Pi^i$ such that

$$E^{(\mu,\pi|\hat{\pi}^i)}(u_i|h) > E^{(\mu,\pi)}(u_i|h),$$

where $\pi|\hat{\pi}^i$ is π with π^i replaced by $\hat{\pi}^i$. A **sequential equilibrium** is a sequentially rational consistent assessment.

4.2 Extensive versus normal form

There are computational aspects of many of the conceptual issues that arise in connection with this model. Perhaps the most important questions, and the first to be addressed historically, involve the comparison of the extensive with the normal form. A **pure strategy** for agent i is a function $s_i : H^i \to A$ with $s_i(h) \in A(h)$ for all h in the domain. Given a pure strategy vector $s = (s_1, \ldots, s_n)$, ρ and π^0 induce a distribution on terminal nodes, and, for a given payoff $u \in \mathbb{R}^{Z \times I}$, a pure strategy vector $s = (s_1, \ldots, s_n)$ consequently induces expected payoffs $u_i(s)$, $i \in I$. This construction is the most common way of passing from an extensive to a normal form game (there are others), and is what is generally understood as "the" normal form of the given extensive game.

Any behavior strategy π^i for agent i induces a normal form mixed strategy $\sigma_i^{\pi^i}$ according to the formula $\sigma_i^{\pi^i}(s_i) = \prod_{h \in H^i} \pi_h^i(s_i(h))$. There are mixed strategies that do not arise in this way, so one can ask whether an agent sacrifices any significant strategic flexibility by using only behavior strategies. Kuhn [1953] showed that, provided the game is one of perfect recall, behavior strategies are strategically adequate in the sense that for any mixed strategy there is a behavior strategy that is **realization equivalent**: regardless of the mixed strategies of the other agents, the mixed strategy and the behavior strategy induce the same distribution on terminal nodes. In particular, a Nash equilibrium in behavior strategies, that is, a vector of behavior strategies such that no agent can increase expected payoff by switching to another behavior strategy, is a true Nash equilibrium in that no agent has an improving deviation in the space of mixed strategies. Conversely, for any mixed strategy Nash equilibrium, a behavior strategy consisting of realization equivalent behavior strategies for the various agents is also a Nash equilibrium.

For agent i the dimension of the set of mixed strategies is $(\prod_{h \in H^i} \#A(h)) - 1$ while the dimension of the set of behavior strategies is $\sum_{h \in H^i}(\#A(h) - 1)$. The general principle that the difficulty of solving an algebraic system is severely affected by the dimension suggests that the ability to work with behavior strategies is an important simplification, and indeed, in the authors' experience, for particular examples the extensive form is generally much easier to solve. In this vein, Wilson [1972] developed a version of the Lemke-Howson algorithm for two person extensive form games. Koller, Megiddo, and von Stengel [1994] show how, for two person games of perfect recall, in the space of sequence strategies (which has the same dimension as the space of behavior strategies) the definition of Nash equilibrium can be expressed as a linear complementary problem, so that the Lemke-Howson algorithm can be applied without first passing to the normal form, which could be much larger.

4.3 Computing sequential equilibria

The behavior strategies of a sequential equilibrium constitute a Nash equilibrium, but there are behavior strategy Nash equilibria that are not sequential, in the sense that there is no consistent belief for which the behavior at unreached information sets is sequentially rational. Sequential equilibrium is generally regarded as better founded conceptually, and we now discuss the computational issues that are particular to it. One important simplification is that, for a game of perfect recall, a consistent assessment is sequentially rational if and only if it satisfies the following weaker condition: the assessment (μ, π) is **myopically rational** if, for each $i \in I$ and $h \in H^i$, there does not exist $\hat{\pi}^i_h \in \Delta(A(h))$ such that

$$E^{(\mu,\pi|\hat{\pi}^i_h)}(u_i|h) > E^{(\mu,\pi)}(u_i|h).$$

(For a proof see KW.) Since $E^{(\mu,\pi|\hat{\pi}^i_h)}(u_i|h)$ is a linear function, it is sufficient that this condition hold with $\hat{\pi}^i_h = a$ for each $a \in A(h)$. Thus we can replace sequential rationality, a nonlinear quantified condition, with a finite collection of unquantified inequalities.

Since unquantified systems are generally easier to solve, it is also significant that consistency can be expressed in an unquantified manner. A **basis** is a set $b = b_A \cup b_X$ where $b_A \subset A$ and $b_X \subset X$. A basis is **consistent** if there is a consistent assessment (μ, π) in which b_A is the set of actions that are assigned positive probability by the various components of π and b_X is the set of nodes that are assigned positive probability by the various components of μ. Kohlberg and Reny [1993] (see also Azhar, McLennan, and Reif [1992] and the Appendix of KW) establish the characterization of consistency expressed in the following two results.

Lemma 3 *b is a consistent basis if and only if: (a) for each $h \in H$ there is at least one $a \in A(h)$ with $a \in b_A$; and (b) there is a function $w : A \to \mathbb{R}_+$ with $w(a) > 0$ if and only if $a \in A - b_A$, such that for each $h \in H$,*

$$b_X \cap h = \operatorname{argmin}_{x \in h} \sum_{\ell=0}^{\ell(t)-1} w(\alpha(p_\ell(t))).$$

Lemma 4 *If b is a consistent basis and π is a behavior strategy such that, for all h and $a \in A(h)$, $\pi_h(a) > 0$ if and only if $a \in b_A$, then (μ, π) is consistent if and only if there is a function $\psi : A \to \mathbb{R}_{++}$ with $\psi(a) = \pi(a)$ for all $a \in b_A$ such that, for all $h \in H$ and $x \in h$*

$$\mu_h(x) = \begin{cases} \frac{\rho(p_{\ell(x)}(x)) \cdot \prod_{\ell=0}^{\ell(x)-1} \psi(\alpha(p_\ell(x)))}{\sum_{x' \in h \cap b_X} \rho(p_{\ell(x')}(x')) \cdot \prod_{\ell=0}^{\ell(x')-1} \psi(\alpha(p_\ell(x')))} & x \in b_X, \\ 0 & x \notin b_X. \end{cases}$$

The conditions stated in Lemma 1 can be expressed as a feasibility problem for a linear program, so the simplex algorithm can be used to determine if a given basis is feasible. For any feasible basis, Lemma 2 parameterizes the associated set of consistent assessments, and since sequential rationality is an algebraic condition without quantifiers, once beliefs are given, we have a characterization of the sequential equilibria for this basis in terms of an unquantified system of algebraic equations and inequalities.

Practical experience in solving games suggests that significant increases in computational efficiency can be obtained by using dominance, and more sophisticated types of reasoning, to eliminate certain bases from consideration without solving the associated algebraic systems. It is intuitively plausible that the beliefs can play an important role in such analysis. In particular examples it is also easy to see how the beliefs can facilitate generalizations of backward induction. For example, if an information set contains two nodes, and the game beginning at this information set is self contained, in that no descendant node is in an information set containing nodes not descended from one of these nodes, then it is natural to solve the part of the game below this information set, treating the beliefs at this information set parametrically, then solve the rest of the game using the derived relation between beliefs at this information set and expected payoffs conditional on each of its two nodes. At this point the efficacy of this approach has not been given formal expression, either in computational theory or in actual software.

It is known that the problem of solving for equilibrium in the extensive form is computationally demanding. Blair and Mutchler [1993] show that, even in a relatively simple class of examples, the computational problems are already NP-hard.

5 Equilbrium Refinements

The ability to compute not just Nash or sequential equilibrium, but also the various refinements that have been proposed in a now quite voluminous literature, would facilitate both application and testing of these concepts. The computational issues associated with these concepts have not been studied in depth. We will first describe the current literature, then discuss the subject in more general terms.

5.1 Two Person Games

Eaves' modification of the Lemke-Howson algorithm to deal with degenerate games is a method of introducing a class of perturbations of the game to make it non-degenerate. As long as the lex-order (the ordering of the columns of the tableau) is chosen appropriately, the algorithm will only terminate at a perfect equilibrium. We now prove this assertion. For this discussion, we use the same notation as that of section 3.1.

Lemma 5 *Eaves' modification of the Lemke-Howson algorithm (as described in section 3.1) will only terminate at a perfect equilibrium.*

Proof: Assume that the algorithm has terminated at the complementary lex-feasible basic solution β. Then, since β is a lex-feasible basis, $T^\beta = [\,-q^\beta \quad A^\beta\,]$ is lex-negative. Let Z be a $2m \times (2m+1)$ dimensional matrix, whose i^{th} column, Z_i, is the basic solution for basis β (that is, components of Z_i corresponding to nonbasis indices are zero) to the equation $A^\beta Z_i = -T_i^\beta$, where, T_i^β is the i^{th} column of T^β. For any $\eta > 0$ define μ_η to be the $2m+1$ dimensional vector

$$\mu_\eta = (1, \eta^1, \eta^2, \ldots, \eta^m, 0, \ldots, 0),$$

and set $z_\eta = Z\mu_\eta$. Write $z_\eta = (y_\eta, x_\eta)$, where y_η and z_η are the first m and last m components of z_η.

The i^{th} row of Z is a vector of zeros if i is not in the basis, so z_η is complementary. If i is in the basis, then the i^{th} row of Z is equal to the negative of the row of T^β containing a 1 in column i, so Z is lex-positive since T^β is lex-negative. Therefore $z_\eta \geq 0$ for sufficiently small η.

In view of the definition of Z we can write

$$[\,-q^\beta \quad A^\beta\,] + A^\beta Z = 0. \tag{5.1}$$

Multiplying this equation through by μ_η yields

$$[\,-q^\beta \quad A^\beta\,]\,\mu_\eta + A^\beta z_\eta = 0.$$

Recalling that $A = [\,U \quad I_m\,]$, set $U^\beta = (B^\beta)^{-1}U$. Then we can rewrite the last equation as

$$U^\beta(y_\eta + \delta_\eta) + (B^\beta)^{-1}x_\eta = q^\beta \qquad \text{or} \qquad U(y_\eta + \delta_\eta) + x_\eta = q.$$

where $\delta = (\eta^1, \ldots, \eta^m)$ consists of the second through $m+1$ entries of μ_η.

But the last equation is just an expression of the restriction that $z_\eta = (y_\eta, x_\eta)$ is a complementary basic feasible solution for a perturbed problem obtained by requiring that each strategy be played with a probability of at least $\eta^j v_{-i}$ (where v_{-i} is the value of the game to the player who does not choose strategy j.) By taking a sequence of such solutions as η goes to zero, we obtain a test sequence that converges to z_0, the lex-feasible basic solution of the original specification of the problem. Hence, z_0 must be a perfect equilibrium. ∎

A stable set, as defined by Kohlberg and Mertens [1988], is (roughly) a set of strategies that is an equilibrium, and which continues to intersect the set of Nash equilibria for any sufficiently small perturbation of the game. Wilson [1992] uses the properties of the lexical version of the Lemke-Howson algorithm to construct an algorithm for computing "**simply stable sets**" of equilibria in finite two-person games. A simply stable set is a weakening of the notion of a stable set defined by Kohlberg and Mertens [1988]. It is a set of strategies that is an equilibrium under a restricted set of perturbations to the game. Wilson's algorithm only finds a sample stable set. However the algorithm is a generalization of the Lemke-Howson algorithm, and like that algorithm, it can be modified to find all 'accessible' stable sets.

5.2 N Person Games

Yamamoto [1993] proposes a homotopy method for computation of a sample proper equilibrium for a general n-person finite game. Since the path of the algorithm is determined by nonlinear equations, there are issues of numerical approximation similar to those raised by Wilson [1971] and Rosenmüller [1971]. Talman and Yang [1994] propose a simplicial subdivision algorithm for finding a sample proper equilibrium of finite n-person games.

Mertens [1988, p. 24] points out that there is in principle a finite procedure for computation of stable sets for n-person games based on a nearly exhaustive procedure based on triangulation of semi-algebraic sets (these are defined in §6), although he does not provide a practical algorithm for implementing this. For extensive form games, McKelvey and Palfrey [1994] propose a homotopy based algorithm for computing a generically unique selection from the set of sequential equilibria for extensive form games.

Turning now to general considerations, perhaps the most important feature presented by the refinement literature is its diversity of methods. While most refinements exclude certain Nash equilibria from the set of solutions, the various notions of stability (Kohlberg and Mertens [1988]) define a solution to be a set of equilibria, so to solve for the set of stable equilibria means finding certain subsets of the set of Nash, or perhaps sequential, equilibria. Some refinements are defined by requiring approximation by certain types of approximate equilibria (Selten [1975], Myerson [1977]). Testing an equilibrium by comparing it, in some sense, with the set of all equilibria (McLennan [1985]), or selected subsets of the set of equilibria (Cho and Kreps [1987], Cho [1987], Banks and Sobel [1987]) is a possible method. Kalai and Samet [1984] ask for equilibria whose supports are, in a certain sense, minimal. Although it has not been much discussed in this literature, it is also possible to use index theory to eliminate some equilibrium, and it is noteworthy that, as is pointed out by Shapley [1974], for nondegenerate games, starting from the extraneous solution the Lemke-Howson algorithm halts at an equilibrium with positive index. (More generally, it always locates an equilibrium with opposite index from the index of the starting point.)

To a very large extent the computability, in principle, of questions associated with these concepts, is a consequence of their being describable as semi-algebraic sets. (See, e.g., Schanuel, Simon and Zame [1989] and Blume and Zame [1994].) Since the various notions of stability are set valued, they present potential counterexamples to this general principle.

6 Finding all Equilibria

For many purposes, an algorithm that yields a single sample equilibrium is unsatisfactory. Even if the resulting equilibrium is perfect, or satisfies some other criterion posed in the literature on refinements of Nash equilibrium, we cannot eliminate the possibility that other equilibria exist and are more salient. Some refinements pose standards that involve comparison of a candidate equilibrium with the other equilibria of the game. Even if (perhaps especially if) one believed that models with multiple equilibria were treacherous, and therefore ill-suited for applications, one would still have an interest in algorithmic methods for determining if there *are* multiple equilibria, or other facts about the *set* of Nash equilibria.

In recent years computer scientists have made a great deal of progress in understanding algorithms that deal with systems of equations and inequalities of multivariate polynomials. The set of Nash equilibria can be represented as such a system, as we explained in section 2, so this material is directly applicable to the task at hand, and the goal of this section will be to give some feeling for the subject by presenting some of the algorithms that appear, at this point, to have the greatest promise. At this point there is no literature concerning how these algorithms might be customized for the particular problems of noncooperative game theory. There is also no literature concerning how these algorithms might effectively utilize knowledge of a sample equilibrium. In the authors' experience, an important idea in organizing the analysis of a game by hand is to find one equilibrium, then ask how other equilibria might differ from this one; there is currently no substantiation of this wisdom in theory or in computational experience.

Generally, the algorithms below are much slower than those discussed earlier, with running times, and in some cases memory requirements, that grow exponentially as various parameters of the input (in particular the dimension) are increased. Exponential algorithms are sometimes loosely described as impractical, and problems for which no better algorithms exist are thought to be intractable, but here these terms seem inappropriate. If all the problems of interest were large in scale, then indeed these algorithms would have little utility, but in fact the simplest games are the ones referred to most frequently in the literature, and many interesting models can be expressed in relatively small trees. While these procedures will not be useful for many problems of interest, they should certainly vastly expand the set of examples for which complete analysis is possible.

To begin with, it is important to recognize that a variety of computational tasks are possible.

1. Determine whether an equilibrium exists or, more commonly, since existence is usually guaranteed, determine whether there exists an equilibrium with some additional property.

2. Determine the dimension of the set of equilibria with some property.

3. In the event that the set of equilibria with some property is 0-dimensional, determine its cardinality.

4. Compute numerical estimates of the equilibria.

5. Determine the topology of the set of equilibria, for instance by presenting a triangulation.

This list is exemplary rather than exhaustive. In fact many refinements of Nash equilibrium depend on the topological relationship between the graph of the best response correspondence and its intersection with the diagonal in $\Sigma \times \Sigma$. Also, some of the tasks subsume others, obviously. Even when the less demanding tasks are not the ultimate goal, they can be useful preparatory steps in more demanding calculations. In particular, computation of numerical values of the various elements of a set of Nash equilibria can be facilitated in several ways by a knowledge of how many equilibria there are to compute.

6.1 Feasibility

Important theorems of pure mathematics show that all the tasks above are, in principle, computationally feasible. A **semi-algebraic set** is a subset $A \subset \mathbb{R}^m$ that is the set of points satisfying a propositional formula $P(x)$ built up from polynomial equations and inequalities in the variables $x_1, \ldots, x_m$, the logical operators 'and', 'or', and 'not', and parentheses. For example $P(x)$ might be the condition '($x_1 \geq 0$ and ${x_1}^2 = {x_2}^3$) or $x_2 < 1$'. As we stressed in Section 2, the set of Nash equilibria is a semi-algebraic set.

There is a more general class of **quantified propositional formulas** of the form

$$P(x) = (Q_1 y_1) \ldots (Q_k y_k) R(x, y)$$

where $Q_1, \ldots, Q_k \in \{\forall, \exists\}$ and $R(x, y)$ is an unquantified propositional formula. In this formula the variables $y_1, \ldots, y_k$ are said to be **bound** by the quantifiers $\forall$ and $\exists$, while the variables $x_1, \ldots, x_m$ are said to be **unbound**. Perfect equilibrium is an example of a solution concept for which all known definitions involve quantified expressions such as "*for all* $\epsilon > 0$ *there exists* $\delta > 0$ such that *for all* trembles ... " The celebrated Tarski-Seidenberg theorem asserts that any quantified propositional formula is equivalent to some unquantified formula, in the sense of determining the same subset of $\mathbb{R}^m$, and in fact the original proof essentially specified an algorithm for generating the unquantified equivalent. This algorithm is obviously impractical, and has never been implemented. More plausible algorithms for quantifier elimination have been developed (cylindrical algebraic decomposition (Collins [1975]) can be adapted to this task), but it has also been shown that the problem is inherently difficult.

It is easily seen that the class of semi-algebraic sets is closed under intersection, union, and complementation. It was long an open problem to show that any such set could be triangulated: Hironaka [1975] was the first to present an acceptable proof. An algorithm developed by Collins [1975], **cylindrical algebraic decomposition**, supplies a more general type of decomposition, from which a triangulation can be derived, so this algorithm constitutes an alternative proof of Hironaka's theorem. These facts are important because the topology of a space with a finite triangulation is completely determined by the finite, combinatoric, data specifying the simplicial complex. Thus a large amount of topological information is, in principle, computable.

6.2 Exemplary Algorithms for semi-algebraic sets

During the last decade the literature in computer science concerning algorithmic analysis of algebraic systems has grown rapidly. There are now many methods that are, at least in principle, applicable to the problems that arise in game theory. In contrast, virtually nothing is known about how one might customize such procedures to take advantage of the special properties of the systems that arise in game theory. (For instance, expected payoffs have degree 0 or degree 1 in each of the strategic probabilities.) Here we will describe two algorithms that currently seem among the most promising from the point of view of the computation of equilibrium.

The **support** of a Nash equilibrium is the set of pure strategies that are assigned positive probability. For a given support the definition of Nash equilibrium can be expressed as a conjunction of polynomial equations (any two strategies in an agent's support have equal expected payoff, probabilities sum to one) and weak inequalities (a strategy for an agent outside his/her support does not have a higher expected payoff than a strategy in the support, probabilities are nonnegative).

We now abstract away from the game-theoretic origins of the problem. Let $p_1, \ldots, p_m$ and $q_1, \ldots, q_k$ be polynomial functions of $x \in \mathbb{R}^m$. A **sign assignment** for $q_1, \ldots, q_k$ is a vector $\sigma = (\sigma_1, \ldots, \sigma_k) \in \{-, 0, +\}^k$. The sign assignment is said to be **satisfied** at a point $x \in \mathbb{R}^m$ if σ_i is the sign of $q_i(x)$, and we let $\sigma(x)$ denote the sign assignment that is satisfied at x. The computational problem studied here is: *determine the number of common roots of $p_1, \ldots, p_m$ satisfying each possible sign assignment for $q_1, \ldots, q_k$.* (We think of $p_1, \ldots, p_m$ as the polynomials used to express the requirements that probabilities sum to one, and that each agent is indifferent between two alternatives that both receive positive probability, while $q_1, \ldots, q_k$ are the polynomials used to express the nonnegativity of probabilities, and the requirement that the expected utility resulting from an unused alternatives does not exceed the expected utility resulting from the alternatives that are used.) A related problem is to determine the set of **consistent** sign assignments for $q_1, \ldots, q_k$; by definition this is $\{ \sigma(x) : x \in \mathbb{R}^m \}$. Note that these problems are unaffected if one of the polynomials is multiplied by a positive rational number.

The algorithm outlined here proceeds in two steps. First, we show how to pass from an instance of the multidimensional problem to a sign assignment problem in which $m = 1$. We then describe the Ben-Or, Kozen, Reif [1986] algorithm for unidimensional sign assignment problems.

6.2.1 The resultant

Let $p_1, \ldots, p_{m+1}$ be homogeneous polynomial functions of $x \in \mathbb{R}^{m+1}$. That is, each p_i is a sum of monomials of the same degree, and in particular, if $p_i(x) = 0$ then, for any $\lambda \in \mathbb{R}$, $p_i(\lambda x) = 0$. For each i, the set of monomials whose coefficients in p_i are allowed to be nonzero is fixed, and we regard the coefficients of p_i as variables, so that each p_i may be viewed as a function of the vector of coefficients a_i. The set of vectors of coefficients $c = (a_1, \ldots, a_{m+1})$ such that $p_1, \ldots, p_{m+1}$ have a common nonzero root (hence a one-dimensional linear subspace of roots) is the set of points satisfying a particular quantified formula, so the Tarski-Seidenberg theorem implies that it can be expressed as the set of points, in the space of the coefficients, satisfying some unquantified formula. In fact (cf. van der Waerden [1950]) the closure of this set of coefficient vectors is the set of roots of a single irreducible (i.e., unfactorable) polynomial, and the **resultant** of $p_1, \ldots, p_{m+1}$ is, by definition, the lowest degree polynomial in the coefficients of $p_1, \ldots, p_{m+1}$ that vanishes precisely on the closure of the set of coefficient vectors for which $p_1, \ldots, p_{m+1}$ have a nonzero root. There is a very familiar example: when each p_i is linear, the resultant is the determinant of the $(m+1) \times (m+1)$-matrix whose i'th row is the vector of coefficients of p_i.

Some methods for computing the resultant have been known for a long time. (See van der Waerden [1949].) These methods are "general" in that they allow all monomials of the same degree as p_i to have a nonzero coefficient in p_i. For most problems of interest the numerical values of at least some of the coefficients are known at the outset. It is possible to compute the relevant resultant symbolically, then substitute the given coefficient values, but for all but the smallest problems this method is too slow and consumes too much memory. Recently a literature in mathematics and computer science has developed around the computation of the **sparse resultant**, which is the polynomial derived from the resultant by setting some of the coefficients to zero. Some of the proposed algorithms also feature methods of taking advantage of prior knowledge of some of the nonzero coefficients. The subject is complex, utilizing surprising and sophisticated methods. It is developing at a rapid pace, and does not seem near resolution, so that although it is a central component of methods for computing game theoretic equilibria, we are unable to treat it in any detail here. For the interested reader we recommend Canny and Emeris [1993], which in many respects represents the current state of the art, and references therein.

Now recall that we began with m polynomial functions $p_1, \ldots, p_m$ on $\mathbb{R}^m$, but more recently we have been considering $m+1$ polynomial $p_1, \ldots, p_{m+1}$ in the variables $x_0, \ldots, x_m$. We pass from the first situation to the second as follows. First, convert the polynomials, originally functions of $x_1, \ldots, x_m$ into homogeneous polynomials by multiplying each monomial by a suitable power of the "homogenizing variable" x_0, where by "suitable" we mean that, after all such multiplications, in each polynomial all monomials have the same total degree. Geometrically this corresponds to the obvious embedding of $\mathbb{R}^m$ as the hyperplane in $\mathbb{R}^{m+1}$ given by the condition $x_0 = 1$, in the sense that the zeros of the derived polynomials in this hyperplane correspond to the zeros of the original polynomials. We will not distinguish notationally between the two versions of $p_1, \ldots, p_m$. Second, we add another linear homogeneous polynomial $p_{m+1} = u_0 x_0 + \ldots + u_m x_m$ to the system.

The resultant of the derived system $p_1, \ldots, p_{m+1}$ is called the **u-resultant** of the given system $p_1, \ldots, p_m$. Taking the coefficients of $p_1, \ldots, p_m$ as given numerically, we regard the u-resultant as a polynomial $R(u_0, \ldots, u_m)$ in $u_0, \ldots, u_m$. Suppose the system $p_1, \ldots, p_m$ (viewed as a system of homogeneous polynomials in the variables $x_0, \ldots, x_m$) has finitely many one dimensional subspaces of solutions, and $\xi^{(1)}, \ldots, \xi^{(q)}$ are nonzero points on these lines. Then, according to our definition, the resultant should vanish if and only if some inner product $\xi^{(\alpha)} \cdot u$ vanishes. Since the resultant is the lowest degree polynomial in the coefficients with this property, it follows that $R = \prod_{\alpha=1}^{r} \xi^{(\alpha)} \cdot u$. Note that $\xi^{(\alpha)}$ corresponds to a root of the original m-variate system $p_1, \ldots, p_m$ if and only if $\xi_0^{(\alpha)} \neq 0$.

The following analysis, due to Canny [1988], is the reduction to a single dimension. The construction begins with the choice of numerical constants $c_1, \ldots, c_m$ such that

(†) for all $\alpha = 1, \ldots, q$, if $\xi_0^{(\alpha)} = 0$, then $\sum_{j=1}^{m} c_j \xi_j^{(\alpha)} \neq 0$.

(‡) for $k = 1, \ldots, m$ and all distinct $\alpha, \beta, \gamma = 1, \ldots, m$,

$$\sum_{i=1}^{m} c_i \xi_i^{(\gamma)} - \xi_k^{(\gamma)} \neq 2 \sum_{i=1}^{m} c_i \xi_i^{(\alpha)} - \sum_{i=1}^{m} c_i \xi_i^{(\beta)} - \xi_k^{(\beta)}.$$

Of course we do not know the points $\xi^{(1)}, \ldots, \xi^{(q)}$ in advance, so we cannot know that a particular choice of $c_1, \ldots, c_m$ is acceptable, but it turns out that a bad choice can be diagnosed at run time, so one can either take a Monte Carlo approach, choosing $c_1, \ldots, c_m$ randomly, which works "with probability one," or one can keep systematic track of which choices fail, so that with enough such "failure" one will be able to solve for $\xi^{(1)}, \ldots, \xi^{(q)}$ by linear algebra.

Form the univariate polynomials

$$p(x) = R(-x, c_1, \ldots, c_m)$$

and

$$t_i^+(x) = R(-x, c_1, \ldots, (c_i + 1), \ldots, c_m),$$
$$t_i^-(x) = R(-x, c_1, \ldots, (c_i - 1), \ldots, c_m),$$

for $i = 1, \ldots, m$.

Recall that $R(u_0, \ldots, u_m)$ factors as the product of the linear forms $u_0 \xi_0^{(\alpha)} + \ldots + u_m \xi_m^{(\alpha)}$. Now if $\xi_0^{(\alpha)} = 0$, then, as a function of x, the corresponding linear factor of p is a constant that is nonzero by (†). Since R is defined only up to multiplication by a nonzero scalar, we may assume without loss of generality that the vectors $(\xi_0^{(\alpha)}, \ldots, \xi_m^{(\alpha)})$ with $\xi_0^{(\alpha)} \neq 0$ have been normalized to have $\xi_0^{(\alpha)} = 1$. The roots of p are then the numbers

$$\theta_\alpha = c_1 \xi_1^{(\alpha)} + \ldots + c_m \xi_m^{(\alpha)}$$

for those α for which $\xi_0^{(\alpha)} \neq 0$.

It is possible that some $t_i^+(x)$ or $t_i^-(x)$ has multiple roots, a situation we wish to avoid. A polynomial is **quadratfrei** if it has no square factors. Any polynomial has a unique factorization as a product of powers of linear factors; the **quadratfrei part** of the polynomial is the product of these factors. As we explain in greater detail below, the Euclidean remainder sequence provides a method of computing all square factors of a univariate polynomial. Let $\hat{t}_i^+(x)$ and $\hat{t}_i^-(x)$ be the quadratfrei parts of $t_i^+(x)$ and $t_i^-(x)$ respectively. From the factorization of R we conclude that the roots of $\hat{t}_i^+(x)$ are the numbers of the form $\theta_\alpha + \xi_i^{(\alpha)}$, while the roots of $\hat{t}_i^-(x)$ are the numbers of the form $\theta_\alpha - \xi_i^{(\alpha)}$. Evidently for given θ the roots of $\hat{t}_i^+(2\theta - x)$ and $\hat{t}_i^-(2\theta - x)$ are the numbers $2\theta - \theta_\alpha - \xi_i^{(\alpha)}$ and $2\theta - \theta_\alpha + \xi_i^{(\alpha)}$ respectively. If $\theta = \theta_\alpha$, then $\hat{t}_i^-(x)$ and $\hat{t}_i^+(2\theta_\alpha - x)$ have the root $\theta_\alpha - \xi_i^{(\alpha)}$ in common, and a little algebra shows that (‡) implies that they cannot have any other common root.

For each $i = 1, \ldots, m$ we form a sequence of polynomials in the variables (θ, x) by setting $z_i^0(\theta, x) = \hat{t}_i^+(2\theta - x)$, $z_i^1(\theta, x) = \hat{t}_i^-(x)$, and, for $j > 1$, setting

$$z_i^j(\theta, x) = w_i^j(\theta) z_i^{j-2}(\theta, x) - y_i^j(\theta) z_i^{j-1}(\theta, x) x^{e_i}$$

where the polynomials $w_i^j(\theta)$, $y_i^j(\theta)$, and the integer e_i are chosen in some way that insures that the degree of z_i^j, as a function of x with coefficients that are polynomials in θ, is less than the degree of z_i^{j-1}. Thus e_i will be the difference between the degrees of z_i^{j-2} and z_i^{j-1}, and w_i^j and y_i^j could be the leading coefficients of z_i^{j-1} and z_i^{j-2} respectively. (In practice one may wish to check whether these leading coefficients have any common factors, in which case w_i^j and y_i^j could be simplified.) For some J, z_i^{J+1} will be a function of θ only. If $\xi_0^{(\alpha)} \neq 0$, then $z_i^{J+1}(\theta_\alpha) = 0$, since $z_i^0(\theta_\alpha, \cdot)$ and $z_i^1(\theta_\alpha, \cdot)$ have a common root, as we saw above. Also, z_i^J must have degree one as a function of x, since otherwise $z_i^0(\theta_\alpha, \cdot)$ and $z_i^1(\theta_\alpha, \cdot)$ would have multiple common roots.

Thus we may write $z_i^J(\theta, x) = d_i(\theta)x + n_i(\theta)$. Since $z_i^J(\theta_\alpha, \theta_\alpha - \xi_i^{(\alpha)}) = 0$ for all α such that $\xi_0^{(\alpha)} \neq 0$, setting

$$r_i(\theta) = \frac{n_i(\theta)}{d_i(\theta)} + \theta,$$

gives a rational function such that $r_i(\theta_\alpha) = \xi_i^{(\alpha)}$. (Note that, while $n_i(\theta)$ and $d_i(\theta)$ may depend on the choices of w_i^j and y_i^j made above, their quotient does not.)

Summarizing, once we have computed p and t_i^+, t_i^-, $i = 1, \ldots, m$, the polynomials d_i and n_i are computed using polynomial remainder sequences, and the rational function r_i is computed according to the formula above. Since p and $r_1, \ldots, r_m$ have the properties indicated at the outset, questions about the possible sign assignments of $q_1, \ldots, q_k$ at such roots can be rephrased as univariate problems by substituting the functions r_i for the various arguments of $q_1, \ldots, q_k$.

6.2.2 Univariate Systems

Now let p and $q_1, \ldots, q_k$ be polynomials in a single real variable x with rational coefficients. Our goal is to find an algorithm determining the number of real roots of p satisfying each possible sign assignment for $q_1, \ldots, q_k$.

For any pair of univariate polynomials f and g, the **Euclidean remainder sequence** $r_0(f, g), \ldots, r_n(f, g)$ is defined by

$$
\begin{array}{lcl}
r_0 & = & f \\
r_1 & = & g \\
r_{i+1} & = & s_i \cdot r_i - r_{i-1} \quad \text{with } \deg r_{i+1} < \deg r_i \quad (i = 1, \ldots, n-1) \\
0 & = & s_n \cdot r_n - r_{n-1}
\end{array}
\tag{6.1}
$$

That is, r_{i+1} is the remainder resulting from division of r_{i-1} by r_i, with r_n being the last nonzero remainder.

For most readers the Euclidean remainder sequence will be familiar as the algorithm for determining the greatest common divisor r_n of f and g. In particular, writing $p = \prod_j (x - \alpha_j)$, where the product is over the roots α_j of p, then differentiating, one can show that p has multiple roots if and only if $\gcd(p, p')$ is a polynomial of positive degree. More generally, if $p = p_1 \cdot {p_2}^2 \cdot \ldots \cdot {p_h}^h$, where each p_ℓ is quadratfrei (has no square factor), then (up to multiplication by a nonzero constant)

$$\gcd(p, p', \ldots, p^{(\ell)}) = p_{\ell+1} \cdot {p_{\ell+2}}^2 \cdot \ldots \cdot {p_h}^{h-\ell},$$

so that

$$p_\ell = \frac{\gcd(p, p', \ldots, p^{(\ell-1)}) \cdot \gcd(p, p', \ldots, p^{(\ell+1)})}{\gcd(p, p', \ldots, p^{(\ell)})^2}.$$

Evidently we have described an algorithm for decomposing p into a product of powers of quadratfrei polynomials. The given problem, determining the number of roots of p with each sign assignment, is evidently solved if we can determine the number of roots of each p_ℓ with each sign assignment, so henceforth we will assume that *p is quadratfrei*. Similarly, we assume that *for each j, p and q_j are relatively prime*, since computation of gcd's allows us to express the solution of the given problem in terms of the solutions of smaller problems for which this is the case.

Consider relatively prime polynomials f and g, and let $r_0 = r_0(f, g), \ldots, r_n = r_n(f, g)$ be the Euclidean remainder sequence. For a point $a \in \mathbb{R}$ at which none of the polynomials in this sequence vanish, let

$$\Sigma(f, g; a) = \#\{\, i = 1, \ldots, n : \operatorname{sign}(r_{i-1}(a)) \neq \operatorname{sign}(r_i(a)) \,\}.$$

For numbers $-\infty \leq a < b \leq \infty$ at which none of the polynomials $r_0, \ldots, r_n$ vanish, let

$$\Delta(f, g; a, b) = \Sigma(f, g; a) - \Sigma(f, g; b)$$

A classical result, known as Sturm's Theorem, states that if p is a quadratfrei polynomial, and $a \leq b$ are numbers that are not roots of $r_0 = r_0(p, p'), \ldots, r_n = r_n(p, p')$, then $\Delta(p, p'; a, b)$ is the number of roots of p between a and b. The Ben-Or, Kozen, Reif (1986) algorithm (hereafter BKR) is based on the following generalization, which they describe as essentially due to Tarski.

Theorem 6 *p, $p' = \frac{dp}{dx}$, and q are relatively prime polynomials, and $a \le b$ are numbers that are not roots of $r_0 = r_0(p, p'q), \dots, r_n = r_n(p, p'q)$, then*

$$\begin{aligned} \Delta(p, p'q; a, b) = \quad & \#\{\, x \in (a,b) : p(x) = 0 \text{ and } q(x) > 0 \,\} \\ & - \#\{\, x \in (a,b) : p(x) = 0 \text{ and } q(x) < 0 \,\}. \end{aligned} \tag{6.2}$$

Proof: To begin with note that (6.2) holds trivially when $a = b$. We argue by considering how each side of (6.2) changes as we pass from $b = c - \epsilon$ to $b = c + \epsilon$, where c is a root of some r_i and there are no other roots of any of the polynomials $r_0, \dots, r_n$ in the interval $[c - \epsilon, c + \epsilon]$. Since p and $p'q$ are relatively prime, r_n is a nonzero constant, so $i = n$ is not a possibility.

Suppose $1 \le i \le n - 1$. Then (6.1) reduces to $r_{i-1}(c) = -r_{i+1}(c)$, so there is exactly one sign change in passing from $r_{i-1}(c \pm \epsilon)$ to $r_i(c \pm \epsilon)$ and then to $r_{i+1}(c \pm \epsilon)$, regardless of the sign of $r_i(c \pm \epsilon)$. Thus the LHS of (6.2) is unaffected by passing from $b = c - \epsilon$ to $b = c + \epsilon$, and since $i \ge 1$ the RHS is also unaffected.

The interesting case occurs when c is a root of $r_0 = p$. To be concrete, suppose that $q(c) > 0$ and $p'(c) > 0$. Then $p'(c)q(c) > 0$, and since $p'(c) > 0$, we have $p(c - \epsilon) < 0 < p(c + \epsilon)$. Therefore

$$\Sigma(p, p'q; c + \epsilon) = \Sigma(p, p'q; c - \epsilon) - 1.$$

We see that if (6.2) holds with $b = c - \epsilon$ then it also holds with $b = c + \epsilon$. The other three possibilities for the signs of $q(c)$ and $p'(c)$ are similar, so the proof is complete. ∎

Fix p and q satisfying the hypotheses of the theorem: p, p', and q are pairwise relatively prime. Let ρ_+ and ρ_- be the numbers of roots of p between a and b at which q is positive and negative, respectively. Let

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Then

$$A \cdot \begin{bmatrix} \rho_+ \\ \rho_- \end{bmatrix} = \begin{bmatrix} \Delta(p, p'; a, b) \\ \Delta(p, p'q; a, b) \end{bmatrix}.$$

Inverting this system yields

$$\begin{aligned} \rho_+ &= \tfrac{1}{2}(\Delta(p, p'; a, b) + \Delta(p, p'q; a, b)), \\ \rho_- &= \tfrac{1}{2}(\Delta(p, p'; a, b) - \Delta(p, p'q; a, b)). \end{aligned} \tag{6.3}$$

To generalize this we introduce a concept from linear algebra. If $B = [b_{hj}]$ is an $m \times m$ matrix and $C = [c_{ik}]$ is an $n \times n$ matrix, then the **Kronecker product** of B and C, denoted by $B \otimes C$, is the $mn \times mn$ matrix whose (hi, jk)-entry is $b_{hj} c_{ik}$. Below we will need the following property of this construct.

Lemma 7 *If B and C are nonsingular square matrices, then $B \otimes C$ is nonsingular.*

Proof: Letting B and C be as described above, suppose $v \in \mathbb{R}^{mn}$ is an element of the kernel of $B \otimes C$. Then, for all h and i,

$$0 = \sum_{j,k} (b_{hj} c_{ik}) v_{jk} = \sum_{j} b_{hj} (\sum_{k} c_{ik} v_{jk}).$$

This means that, for each i, the m-vector with j^{th} component $\sum_k c_{ik} v_{jk}$ is in the kernel of B and therefore 0. But then we see that for each j, $(v_{j1}, \ldots, v_{jm})$ is in the kernel of C. Thus $v = 0$. ∎

Let p be a quadratfrei polynomial, and let $q_1, \ldots, q_k$ be polynomials, each of which has no factor in common with either p or p'. Fix $a < b$, where these numbers are not roots of $p, p', q_1, \ldots, q_k$. Let $\sigma^1, \ldots, \sigma^r$ be the sign assignments that are satisfied by at least one root of p between a and b, and let $\rho_{\sigma^1}, \ldots, \rho_{\sigma^r}$ be the corresponding numbers of such roots. Let $\pi^1, \ldots, \pi^r$ be a collection of products of the form ${q_1}^{e_1} \cdot \ldots \cdot {q_k}^{e_k}$ in which each e_i is either 0 or 1. Finally let Z be the $r \times r$ matrix whose entry Z_{ij} is 1 or -1 according to whether the sign of π^j is positive whenever sign assignment σ_i is satisfied. Then

$$Z \cdot \begin{bmatrix} \rho_{\sigma^1} \\ \vdots \\ \rho_{\sigma^r} \end{bmatrix} = \begin{bmatrix} \Delta(p, p'\pi^1; a, b) \\ \vdots \\ \Delta(p, p'\pi^r; a, b) \end{bmatrix} \tag{6.4}$$

If q_{k+1} is another polynomial that has no factor in common with p or p', then (thinking of $Z \otimes A$ as obtained by replacing each entry of Z with a 2×2 cell containing A multiplied by that entry) the Theorem implies that

$$(Z \otimes A) \cdot \begin{bmatrix} \rho_{\sigma^1+} \\ \rho_{\sigma^1-} \\ \vdots \\ \rho_{\sigma^r+} \\ \rho_{\sigma^r-} \end{bmatrix} = \begin{bmatrix} \Delta(p, p'\pi^1; a, b) \\ \Delta(p, p'\pi^1 q_{k+1}; a, b) \\ \vdots \\ \Delta(p, p'\pi^r; a, b) \\ \Delta(p, p'\pi^r q_{k+1}; a, b) \end{bmatrix} \tag{6.5}$$

where (for example) σ^1+ denotes the sign assignment for $q_1, \ldots, q_{k+1}$ obtained by appending a 1 to σ^1.

An algorithm for computing the number of roots of p satisfying each possible sign assignment for $q_1, \ldots, q_k$ is now apparent. Let Z above be the k-fold Kronecker product of the matrix A, compute the numbers $\Delta(p, p'\pi^j; a, b)$ for the $r = 2^k$ products π^j, and solve (†) for the numbers ρ_{σ^i}. Since Z is a $2^k \times 2^k$ matrix, the running time of this method is evidently exponential in k. However, by performing the calculation in an iterative fashion, adding the polynomials q_i one at a time and keeping track only of the sign assignments that are actually satisfied by roots of p between a and b, this can be drastically improved. More precisely, we have the following inductive step of the algorithm:

1. Given an invertible Z_k satisfying (6.4) above, let $Z''_{k+1} = Z_k \otimes A$.

2. Solve equation (6.5) above for the numbers of roots of p between a and b satisfying each sign assignment.

3. Eliminate the zeros of the vector of numbers $\rho_{\sigma^i\pm}$, and let Z'_{k+1} be obtained from Z''_{k+1} by eliminating the corresponding columns.

4. Let Z_{k+1} be a square matrix consisting of a maximal collection of linearly independent rows of Z'_{k+1}. Eliminate the components of the vector of numbers $\Delta(p, p'\pi^r; a, b)$ and $\Delta(p, p'\pi^r q_{k+1}; a, b)$ that do not correspond to the rows of Z_{k+1}.

With this modification, the dimension of Z_k cannot exceed the number of roots of p, which of course is bounded by the degree of p, so for given p the running time is linear in k.

This completes the description of the univariate component of the BKR algorithm. They point out that, for the particular problem of determining the set of consistent sign assignments, there is a multidimensional extension. First, observe that we have effectively described an algorithm for determining the set or consistent sign assignments for a collection $p_1, \ldots, p_k$ of quadratfrei, pairwise relatively prime polynomials, since for each p_i we may apply the procedure above with p_i in the role of p. Now suppose that $p_1, \ldots, p_k$ are polynomials in $x_1, \ldots, x_m$, and consider the particular problem of determining the set of sign assignments satisfied by the various points in $\mathbb{R}^m$. We treat each polynomial in this sequence as a univariate polynomial, in the single variable x_m, with coefficients in the field of rational functions in the variables $x_1, \ldots, x_{m-1}$. Then each point in $\mathbb{R}^{m-1}$ determines a collection of univariate polynomials in x_m obtained by evaluating the coefficients, for which some collection of sign assignments are possible, and the sign assignments that are consistent for $p_1, \ldots, p_k$ will be the ones obtained in this way as $x_1, \ldots, x_{m-1}$ vary over $\mathbb{R}^{m-1}$.

Now observe that the Euclidean remainder sequence is defined for univariate polynomials over *any* field of coefficients, including the field of rational functions in $x_1, \ldots, x_{m-1}$. The consistent sign assignments for $p_1, \ldots, p_k$ will be those allowed by the sign assignments of the polynomials, in the variables $x_1, \ldots, x_{m-1}$, that arise in the relevant Euclidean remainder sequences computed with respect to x_m, then specialized to $x_m = a = -\infty$ and $x_m = b = \infty$. (That is, we need only the signs of the leading coefficients.) In short, given a collection of polynomials in m variables, there is a method of passing to a collection of polynomials in $m-1$ variables whose consistent sign assignments determine the consistent sign assignments of the given system.

The complexity of this calculation grows very rapidly as the number of variables increases. An important idea in understanding why this is so is the general principle that symbolic operations are expensive in comparison with the corresponding calculations in which symbolic variables are replaced by numerical values as soon as possible. In contrast with the method based on the u-resultant, the multidimensional BKR algorithm offers very little opportunity for "specialization" of variables before the inductive descent reaches the univariate case.

6.2.3 Numerical computation of the solutions

In view of the reduction to one dimension presented above, for the purposes of numerical computation it is important to compute real roots of a polynomial $p(x)$ of one variable, which simultaneously satisfy a set of inequality constraints $q_i(x) \geq 0$. In the following discussion we will assume that the number of such roots has already been determined. In particular, this gives a stopping rule.

There are a number of ways to approach the above problem. One is to find *all* roots of p, and check each to see if it satisfies the constraint. To find all roots, we can repeatedly apply Sturm's theorem to p until we obtain an interval containing just the leftmost remaining root, and then apply any of a standard set of line search methods to find the root of p in the interval. We can inductively apply this procedure until all roots have been found. For each root, we must check whether it satisfies the constraints $q_i(x) \geq 0$. Since we only obtain an approximation to the root, the inequality constraints may be difficult to verify by function evaluation. Thus another application of BKR may be necessary to verify that the roots found satisfy the constraints.

An alternative procedure would be to successively add sets of linear equations to Γ, applying BKR to the augmented set, until we reach a point where each consistent sign assignment of the set of polynomials is contained in a unique interval. Then a line search algorithm can be applied to each interval containing a consistent sign assignment

There are also a number of algorithms which will find all roots of a system of polynomials directly. See, *e.g.*, Ben-Or, Feig, Kozen, and Tiwari [1988], Drexler [1978], Garcia and Zangwill [1979, 1980], and Huber and Sturmfels [1993].

6.3 Complexity of finding game theoretic equilibria

The domain of practical applicability of the algorithms described above, and of related algorithms, is largely determined by the rate at which the time and/or memory requirements of the calculation grow as the 'size' of the input grows. For many of the algebraic algorithms that deal with unquantified systems the time complexity is bounded above by functions that grow exponentially as the dimension of the problem increases. For a fixed dimension the time requirements are bounded by polynomial functions in appropriate measures of the size of the inputs. Reflecting on the fact that $\mathbb{R}^m$ has 2^m orthants, it seems unlikely that one can do much better than this for general algebraic computations.

These considerations leave open the possibility that there might be faster algorithms specifically tailored to deal with the algebraic systems arising in game theory. For large classes of problems a lower bound on the complexity is given by the size of the desired output. This line of reasoning gives a particular theoretical significance to the already interesting question of how many Nash equilibria a game can possess.

Relatively little is known theoretically about the number of Nash equilibria of a game. Since the Nash equilibria are the fixed points of a continuous function, the theory of the fixed point index allows one to assign an integer called the **index** to each set of Nash equilibria that is both open and closed in the relative topology of the set of fixed points. A crucial property of the index is *additivity*: the index of a union of two disjoint 'relatively clopen' sets is the sum of the indices of the two sets. Since the set of Nash equilibria is a semi-algebraic set, and any semi-algebraic set is a union of finitely many path-connected components, the index is determined by the indices of the components. The general theory of the index implies that the indices assigned to the elements of any partition of the set of fixed points must sum to one. It is known that for a finite normal form game, generically in the space of payoffs, each equilibrium is isolated, and its index is either 1 or -1, so that the the number of Nash equilibria is odd. Gul, Pearce and Stachetti [1991] point out that a strict pure equilibrium necessarily has index 1, so that for a generic finite normal form game with $2\alpha + 1$ Nash equilibria, since all equilibria are strict, at least α are non-degenerate mixed strategy equilibria.

If we are searching for all equilibria, and have found some number of Nash equilibria, these results may sometimes give us information that there are Nash equilibria which have not yet been found. Thus, if we have found an even number of Nash equilibria in a generic game, we know there must be at least one more. If we have found k pure strategy equilibria in a generic game, then we know there must be at least $k - 1$ non-degenerate mixed equilibria. However, these results are not useful in informing us when we have found all Nash equilibria, and hence cannot be used to give us a stopping rule in any numerical computational of Nash equilibria.

The total number of equilibria for a game is the sum over all possible supports of the number of totally mixed equilibria for that support. So one approach to finding the number of equilibria would be to study the number of totally mixed equilibria on each support. Of course, in practice, we may not be dealing with a game in general position. So the number of equilibria for a given game could be infinite. In the case that we are not dealing with a generic game, a more tractable question may be to find the number of **regular equilibria**. Regular equilibria are (roughly) Nash equilibria in which unused strategies have strictly suboptimal payoffs, and in which the derivative of the payoff function at the equilibrium is of full rank.

Pick an arbitrary $s \in S$. Set $D_i = S_i - \{s_i\}$, and $D = \cup_{i \in N} D_i$. Let Φ be the number of permutations of D that do not map any element of any D_i into D_i, and let $\phi_i = |D_i|!$ be the number of permutations of D_i. McKelvey and McLennan [1994] prove that the maximum number of completely mixed regular Nash equilibria is $\mathcal{N}(S) = \Phi / \prod_i \phi_i$. They also prove that this is a tight bound by showing it is possible to construct games that achieve the bound. A similar computation gives an upper bound on the number of regular Nash equilibria for any given support.

For an n-person game with $n > 2$, usually $\mathcal{N}(S) > 1$. For example, in an n-person game where each player has 2 pure strategies, $\mathcal{N}(S)$ is equal to the number of **derangements** of the integers from one to n, where a derangement is defined as a permutation with no fixed point. In this case $\mathcal{N}(S)$ is given by the recursive formula $A_n = (n - 1) \cdot [A_{n-1} + A_{n-2}]$, where $A_1 = 0$, and $A_2 = 1$. In a five person game, this is 44, and in a 10 person game, this is 1,334,961.

Let $k = \min |S_i|$ be the size of the smallest strategy space. For fixed n, $\mathcal{N}(S)$ is exponential in k and for fixed k, $\mathcal{N}(S)$ is exponential in n. Since $\mathcal{N}(S)$ is the number of totally mixed regular Nash equilibria, the number of Nash equilibria must be at least as large as $\mathcal{N}(S)$. Thus, even if algorithms could be constructed whose complexity is linear in these parameters, it follows from the above result that the worst case computational complexity of finding all Nash equilibria is at least exponential in n and k.

For a two-person game, $\mathcal{N}(S) = 1$. A two person game will have at most one regular equilibrium per support. Hence the maximum number of regular equilibria is no greater than $(2^{m_1} - 1) \cdot (2^{m_2} - 1)$. On the other hand it is easy to construct games with $(2^{m^*} - 1)$ regular Nash equilibria, where $m^* = min\ [m_1, m_2]$: Assume $m_1 = m^*$ and set $u_i(s_{1j}, s_{2k})$ $= 0$ if $j \neq k$, and $u_i(s_{1j}, s_{2k}) = a_j > 0$ if $j = k$. Then for any non-empty support $C \subseteq S_1$, $p^C = (p_1^C, p_2^C)$ is a regular Nash equilibrium if

$$p_{ij}^C = \prod_{k \in C - \{j\}} a_k / (\sum_{l \in C} \prod_{k \in C - \{l\}} a_k) \text{ when } j \in C,$$

$$p_{ij}^C = 0 \text{ otherwise.}$$

Thus, the maximal number of regular equilibria for a two person game is somewhere between $(2^{m^*} - 1)$ and $(2^{m_1} - 1) \cdot (2^{m_2} - 1)$. In any case, it is clear that the maximum number of regular Nash equilibria is exponential in the minimal number of strategies.

The above complexity calculations represent worst case situations. We might hope that an "average" game would be better behaved than the worst case game. We do not know the answer to this.

6.3.1 Two Person Games

In the case of two person games, each of the equations and inequalities in the definition of Nash equilibrium is linear. So the problem can be re-formulated as a linear complementarity problem. Further, for any support the set of equilibria with that support is a convex set (possibly empty), and its closure is the convex hull of its extreme points. There are a finite number of extreme points, each of which corresponds to a set of k equations and inequalities, whose matrix of coefficients is of full rank, which yield a feasible solution. It follows that a "brute force" method of finding all solutions is explicit enumeration: For each possible support, check for a feasible solution for [5]-[6]. If there is one, either it is unique, or we can find it's finite set of extreme points, whose convex hull represents the set of Nash equilibria for that support.

No genericity assumption is required in the above algorithm. This procedure will locate all equilibria even if there are an infinite set of equilibria. Such a procedure was first suggested by Mangasarian [1964].

Since the number of possible supports for an n-person game is $\prod_{i \in N}(2^{m_i} - 1)$, it is clear that even in a two person game, the above method has computational complexity that is at least exponential in the maximum size of the strategy spaces. Thus we would not expect this method to be feasible on large games. However, this procedure has been implemented by Dikhaut and Kaplan [1991] in Mathematica, and by the first author in C. The authors' experience indicates one can solve an 8×8 game in approximately 30 seconds on a 486/66MH machine, and games up to 12×12 are feasible. One would expect that more sophisticated implementations, which take account of the dominance structure of the game to eliminate whole sets of supports, would substantially improve such algorithms, at least on average games. Koller and Megiddo [1994] give an algorithm for finding all equilibria of a two person extensive form game that runs in time that is exponential in the size of the extensive form.

6.3.2 N-Person Games

For games with more than two players, even if the input data are rational, an isolated Nash equilibrium need not be rational (see *e.g.* Nash [1951] *p.* 294 for an example). Second, the set of equilibria with a given support need no longer be a convex, or even connected set, as the following example illustrates. In this example, $n = 3$, $S_i = \{s_{i1}, s_{i2}\}$ for $i \in N$. The payoff function is given in the following table:

s_{31}	s_{21}	s_{22}	s_{32}	s_{21}	s_{22}
s_{11}	$(9,8,12)$	$(0,0,0)$	s_{11}	$(0,0,0)$	$(3,4,6)$
s_{12}	$(0,0,0)$	$(9,8,2)$	s_{12}	$(3,4,4)$	$(0,0,0)$

A mixed strategy $p = (p_1, p_2, p_3) \in \Delta$ is of the form $p_1 = (p, 1-p)$, $p_2 = (q, 1-q)$, $p_3 = (r, 1-r)$, for some $p, q, r \in [0,1]$. We abbreviate p by (p,q,r). Then any Nash equilibrium to the game with full support must satisfy the equations

$$\begin{aligned}
9qr + 3(1-q)(1-r) &= 3q(1-r) + 9(1-q)r \\
8pr + 4(1-p)(1-r) &= 4p(1-r) + 8(1-p)r \\
12pq + 2(1-p)(1-q) &= 6p(1-q) + 4(1-p)q
\end{aligned}$$

Collecting terms and factoring, this becomes

$$\begin{aligned}
(6q-3)(4r-1) &= 0 \\
(12r-4)(2p-1) &= 0 \\
(8p-2)(3q-1) &= 0
\end{aligned}$$

This system of equations has exactly two solutions, one at $(p,q,r) = (\frac{1}{4}, \frac{1}{2}, \frac{1}{3})$, and one at $(p,q,r) = (\frac{1}{2}, \frac{1}{3}, \frac{1}{4})$. (This game has a total of nine equilibria: four pure strategy equilibria, three equilibria where two players mix and the other adopts a pure strategy, and two equilibria with full support.) It should be noted that the equilibria in this example are all regular Nash equilibria, and this is a generic example, in the sense that any small perturbation of the payoffs in the game will yield a game which also has nine Nash equilibria, two of which are distinct and isolated equilibria with full support.

So it is clear that the computational problem for the n-person case is substantially more difficult than the 2-person case.

7 Practical Computational Issues

In the above sections, we have considered the problem of computation of Nash equilibria primarily from a theoretical point of view. From the point of view of practical application, there are a number of questions that need to be addressed.

7.1 Software

Many of the algorithms we have discussed are quite complicated. Implementation of the algorithms in computer code involves issues of numerical stability, which we have not discussed, as well as questions of efficiency. One impediment to the routine use of the methods we have discussed has been the lack of generally available software that implements the algorithms. For extensive form games, this problem is particularly acute, since every extensive form game is different. If it were necessary for a researcher to select and implement their own version of a solution algorithm for each game that they encounter, then one would not expect that these methods would be applied very widely.

The authors have been involved in development of a computer software package (GAMBIT) to address the above problem. An early version of the software was developed by the first author.[2] Both authors together with Ted Turocy are now involved in making major revisions to GAMBIT.

7.1.1 Current version of GAMBIT

The original version of GAMBIT is a program that allows for the interactive building and solving of extensive form games. The user sees on the computer terminal a graphics display of the current extensive form game, and is able to navigate around the extensive form using commands from the keyboard. A series of editing options allows the user to alter the existing tree by adding, inserting, changing or deleting portions of the existing extensive form. Thus, at a given node, the user may choose to insert a new node, change the player who has control of that node, change the information set to which the node belongs, add or delete a branch from that node, delete the node, copy that node to another part of the tree, or attach an outcome, with payoffs to each of the players, to that node. After any change, the program checks for consistency of the new extensive form, and then redraws the tree if the changes are legal. In this manner, GAMBIT allows for the entry of any valid finite extensive form game. Games of incomplete information can be entered by treating the type distribution as an initial chance move, and specifying player information sets so that each player knows only his/her own type. Stochastic games can be dealt with by having several game elements, and having one game be the outcome of another game. Infinitely repeated games with discounting can be dealt with in a similar manner.

[2] available via anonymous *ftp* at *hss.caltech.edu* in the directories *pub/rdm/gambit* (DOS version) and *pub/rdm/xgambit* (UNIX version.)

For any extensive form game, GAMBIT can construct the corresponding normal form (either the reduced, or full normal form). A number of algorithms for finding equilibria for the normal form game are available. The normal form equilibrium strategies that are found are then converted back to behavioral strategies to obtain solutions to the extensive form.

7.1.2 Revisions to GAMBIT

We are currently making extensive revisions to GAMBIT to add several features: a command language, a modular structure, and support for additional algorithms described in section 6 for computing all equilibria. The most significant feature from the point of general application is the GAMBIT command language (GCL).

The basic philosophy behind the changes is to provide both a programming environment and a computer language to make it easy for both programmers and applied researchers to manipulate and do operations on extensive and normal form games.

Programming Environment

Regarding the programming environment, a programmer who wants to implement an algorithm for finding a particular solution or refinement should only have to worry about writing the code for the algorithm, and not have to worry about writing code to build up or manipulate the game that is to be solved. For example, prior to executing any algorithm, in addition to building up the game, one would typically want to iteratively eliminate either weakly or strongly dominated strategies, and only invoke the algorithm on the reduced game.

To this end, the computer code is being completely rewritten in C^{++} to make it modular and allow a standardized interface to the extensive and normal form. Most algorithms which operate on normal form games need access to the normal form only to evaluate the payoff at a given mixed strategy profile. Similarly, most algorithms that operate on the extensive form could work either directly off of the agent normal form, or only need access to the extensive form to obtain evaluation of certain attributes of the extensive form (such as the payoff or the beliefs at an information set) for a given mixed behavioral strategy profile. Thus, by providing standard data structures for mixed strategy profiles and behavior strategy profiles and a standard interface to obtain the necessary function evaluations, this would enable a programmer to write such algorithms easily. By simply linking in the relevant normal and extensive form libraries, a programmer could write a standalone program that would have available all of the functionality for building and manipulating the game forms without the programmer having to know or have to deal with the internal workings of this code. Alternatively, the same code could be linked into GAMBIT or the GCL to make it available within those platforms.

Command Language

Similar issues arise from the point of view of an applied researcher, trying to make use of existing code developed by others. Here the problem is that code must be available in a way that it can be used by individuals without any particular programming expertise, but in a way that allows researchers to address questions that are unique to their own particular application. To some extent a program like GAMBIT provides such an environment, since it allows easy building and analysis of extensive form games in an interactive setting. However, while an interactive environment is useful for some applications, it requires constant user interaction, and hence is not suitable for many potential uses.

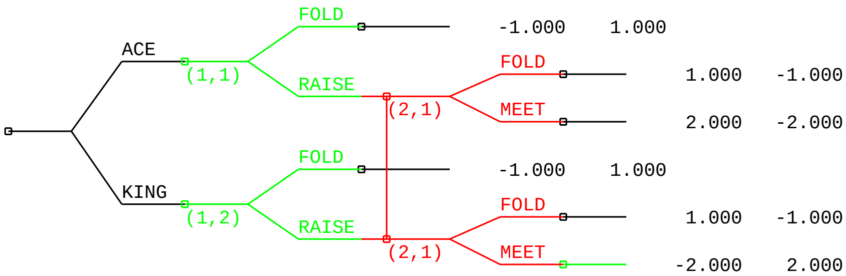
For many applications, it is necessary to solve a game repeatedly with different values of the parameters, to be able to do intermediate calculations on the results (perhaps conditional on what one has found out so far), and to have a record of what has been done. This is particularly true in econometric applications or in certain theoretical applications such as the search for counterexamples. For such usage, an interactive environment is unsuitable. To facilitate such usage, we are currently (with Ted Turocy) writing a command language for GAMBIT. This is a project that is currently in progress. So the following description is intended only to give a flavor for the intended final product.

The command language is a language with Mathematica style syntax. The language contains a number of data types, some corresponding to standard numerical data types (int, double, rational) and some corresponding to elements of games (NormalForm, ExtensiveForm, Node, InformationSet, Player, Outcome, etc.) Variables can be assigned to take on any of these data types, and these variables can then be operated on by an array of functions. Each function has a number of required arguments, (assigned by the operator "$->$") as well as some possible optional arguments.

Some functions are useful for building up extensive form games. These commands each have natural analogues to functions that can be performed (more easily if one only has to do it once) in the interactive version of GAMBIT. For example

```
efg := NewEfg[];
root := RootNode[e-> efg];
AppendNode[n->root,pl->0,br->2];
AppendNode[n->root#1,pl->1,br->2];
AppendNode[n->root#2,pl->1,br->2];
AppendNode[n->root#1#2,pl->2,br->2];
AppendNode[n->root#2#2,pl->2,br->2];
MergeInfosets[n1->root#1#2,n2->root#2#2];
SetActionProbs[n->root,probs->{.5,.5}];
```

is a sequence of commands to create the extensive form tree for a two person simultaneous move game of poker, illustrated in figure 1. It defines the variable "root" to be the root node of a new extensive form game, then creates a decision node for player 0 (chance) at the root node, creates a decision node for player 2 at the first branch of the root node (root#1), does the same at the second branch of the root node, and then merges the two nodes for player 2 into one information set.
Labels and outcomes could then be attached to nodes of the tree by



A simple poker game

```
SetActionNames[n->root,name->{''ACE'', ''KING''}];
     .
     .

AttachOutcome[n->root#1#1,outc->1];\\
SetOutcome[outc->1,value->{-1, 1}];
     .
     .
```

Other functions are useful for manipulation and solution of games. For example

```
nfg := ExtToNorm[e->efg];
AllLemke[n->nfg];
```

converts the extensive form game "efg" to a normal form game, as in the following table, and then finds all solutions that are accessible via the Lemke-Howson algorithm. (In the normal form, the strategies are labeled by the action taken at each information set, so "RF" means that Player 1 Raises with an Ace and Folds with a King). This game has a unique Nash equilibrium at $p_1 = (0, 0, 2/3, 1/3)$, and $p_2 = (1/3, 2/3)$.

	F	M
FF	(-1, 1)	(-1,1)
FR	(0,0)	(-1.5,1.5)
RF	(0,0)	(.5,-.5)
RR	(1,-1)	(0,0)

Table 1
Normal form for poker game

The GCL allows lists of any data type, and has flow control statements, (While, If, For) which have similar functionality and syntax as those in Mathematica. So

```
list := {``e01.nfg'', ``e02.nfg'', ``e03.nfg''}
For[i := 1, i <= Length[l->list], i := i + 1,
   nfg := ReadNfgFile{file->list[[i]];
   While[ElimStrongDom[n-> nfg],DeleteDom[n <-> nfg]];
   SimpDiv[n->nfg];
];
```

would process three files. For each one it would first successively eliminate strongly dominated strategies, and then apply the simplicial subdivision algorithm to the reduced game.

7.2 Computational Complexity

We illustrated the functionality of GAMBIT and the GCL with a very simple example, which can easily be solved by hand. Only slightly larger examples quickly grow in complexity to where they are very difficult or impossible to solve by hand. Computer programs such as GAMBIT are a useful tool in helping to analyze and solve such games. For example, the authors have found these programs indispensible in the design, solution and subsequent econometric analysis of game theory experiments (El-Gamal, McKelvey and Palfrey [1993] and McKelvey and Palfrey[1992, 1993, 1994]). While these applications illustrate that there is a domain of problems for which the algorithms discussed in this survey are useful, they do not indicate how large the domain is. In this section, we discuss briefly this question.

Computational complexity considerations suggest that the worst case performance of all of the algorithms we have discussed is at least exponential in the size of the problem, so that the methods are inherently constrained in the scope of applicability to problems of practical interest. However, the worst case performance may not be the correct measure if problems that arise in practice tend to be more well behaved. An more important question is what is the average case complexity?

Unfortunately, there does not seem to be any systematic evaluation in the literature of the algorithms we have discussed in terms of their performance on average games. Part of the problem is that it is not clear what an average game is.

To get a rough indication of the size of game that is soluble, we generated a set of random normal form games. For each size game reported, we generated 100 random games, and compared some of the algorithms on these games. In all cases, we just searched for *one* equilibrium.

For two person games, the Lemke-Howson algorithm clearly outperforms the other algorithms tested (as currently implemented in GAMBIT). We also tested a simplicial subdivision with restart, and a function minimization algorithm. Table 2 gives information for the Lemke-Howson algorithm, on these random games.

k	Number Pivots	Total Time
2	2.74	.0200
3	3.84	.0156
4	4.56	.0198
6	6.46	.0271
8	7.66	.0383
12	13.16	.0842
16	19.23	.1737
24	33.87	.5772
32	78.17	2.153
48	210.4	12.22
64	426.9	43.68
96	819.2	182.1

Table 2

Performance of Lemke Howson on 100 random games[3]
(k = number of strategies per player)

Based on the data in Table 2, it appears that the speed of the Lemke-Howson algorithm, at least in this range, is approximately polynomial in the size, k, of the strategy space.

We have not performed similar comparisons on n-person games. Undoubtedly they will be considerably slower, since the Lemke-Howson algorithm is no longer available.

The above timings reflect the time required to solve a normal form of a given size. There is at least one consideration that would ease the computational burden for a specific application. Namely, there is usually some pre-processing that can substantially reduce the size of the game before application of any solution algorithms. In the case of extensive form games, if there are non-trivial subgames, then the subgames can be recursively solved, and from a computational point of view, the important variable is the maximum size of a subgame. We have seen also that solving directly on the extensive form can lead to substantial reduction of the strategy space over the normal form. However, even if one moves to the normal form, it is important to note that one should use the reduced normal form, which is typically much smaller than the normal form. Finally, on both normal and extensive form games, one can successively eliminate strongly dominated strategies without eliminating any Nash equilibria. If one wants only to find a sample equilibrium, then one can successively eliminate weakly dominated strategies, and any equilibrium to the reduced game will be an equilibrium to the original game. In short, the effective size of the strategy space for which one has to apply any solution algorithm is frequently much smaller than the original size of the strategy space.

[3]Time in seconds on a SUN 4/ELC

8 References

Azhar, S., A. McLennan, and J. H. Reif, "Computation of equilibria in noncooperative games," mimeo, University of Minnesota, (1992)

Banks, J. S., and J. Sobel, "Equilibrium selection in signalling games," *Econometrica*, 55 (1987): 647-661.

Ben-Or, M., D. Kozen, and J. Reif, "The complexity of elementary algebra and geometry," *Journal of Computer and System Sciences*, 32 (1986): 251-264.

Ben-Or, M., E. Feig, D. Kozen, and P. Tiwari, "A fast parallel algorithm for determining all roots of a polynomial with real roots," *SIAM Journal of Computation*, 17 (1988): 1081-1092.

Blair, J. R. S., and D. Mutchler, "Pure strategy equilibria in the presence of imperfect information: NP-hard problems," University of Tennessee, Dept. Computer Science, (December 1993).

Blume, L. and W. R. Zame, "The algebraic geometry of perfect and sequential equilibrium," *Econometrica*, 62 (1994): 783-794.

Brown, G. W., and J. von Neumann, "Solutions of Games by Differential Equations," in *Contributions to the Theory of Games*, H. W. Kuhn and A. W. Tucker (eds.), Annals of Mathematical Studies Number 24, (Princeton University Press, Princeton: 1950):73-79.

Canny, J., "Some algebraic and geometric computations in PSPACE," *ACM Symposium on Theory of Computing*, (1988): 460-467.

Canny, J., and I. Emeris, "An efficient algorithm for the sparse mixed resultant," *Proceedings, AAEEC*, (1993): 89-104.

Cho. I., "A refinement of sequential equilibrium," *Econometrica* 55 (1987): 1367-1389.

Cho. I., and D. M. Kreps, "Signalling games and stable equilibrium," *Quarterly Journal of Economics* 102 (1987): 179-221.

Collins, G. E., "Quantifier elimination for real closed fields by cylindrical algebraic decomposition," in *Second Gl Conference on Automata Theory and Formal Languages*, vol. 33 of Lecture Notes in Computer Science, (Berlin: Springer-Verlag, 1975): 134-183.

Dickhaut, J., and T. Kaplan, "A program for finding Nash equilibria," *The Mathematica Journal*, 1 (1991): 87-93.

Doup, T. M., "Simplicial Algorithms on the Simplotope," *Lecture Notes in Economics and Mathematical Systems*, #318 (Berlin: Springer-Verlag, 1988).

Doup, T. M., and A. J. J. Talman, "A new simplicial variable dimension algorithm to find equilibria on the product space of unit simplices," *Mathematical Programming*, 37 (1987a): 319-355.

Doup, T. M., and A. J. J. Talman, "A continuous deformation algorithm on the product space of unit simplices," *Mathematics of Operations Research*, 12 (1987b): 485-521.

Drexler, F. J., "A homotopy method for the calculation of all zeros of polynomial ideals," in *Continuation Methods*, ed., H. Wacker, (New York: Academic Press, 1978).

Eaves, B. C., "The linear complementarity problem," *Management Science*, 17 (1971): 612-634.

Eaves, B. C., "Homotopies for computation of fixed points," *Mathematical Programming*, 3 (1972): 1-22.

El-Gamal, M., R. D. McKelvey, and T. Palfrey, "A Bayesian sequential experimental study of learning in games," *Journal of the American Statistical Society*, 42 (1993): 428-435.

van den Elzen, A. H., and A. F. F. Talman, "Finding a Nash equilibrium in noncooperative n-person games by solving a sequence of linear stationary point problems," *ZOR-methods and Models of Operations Research*, 35 (1994): 27-43.

Garcia, C. B., C. E. Lemke, and H. Luethi, "Simplicial approximation of an equilibrium point for non-cooperative n-person games," in *Mathematical Programming*, T. C. Hu and S. M. Robinson, eds, (New York: Academic Press, 1973): 227-260.

Garcia, C. B., and W. I. Zangwill, "Finding all solutions to polynomial systems and other systems of equations," *Mathematical Programming*, 16 (1979): 159-76.

Garcia, C. B., and W. I. Zangwill, "Global calculation methods for finding all solutions to polynomial systems of equations in n variables," in *Extremal Methods and Systems Analysis*, (Heidelberg New York: Springer-Verlag, 1980).

Gul, F., D. Pearce, and E. Stacchetti, "A bound on the proportion of pure strategy equilibria in generic games," mimeo, Stanford University (1991).

Harker, P. T., and J. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications," *Mathematical Programming*, 48 (1990): 161-220.

Hironaka, H., "Triangulation of algebraic sets," *AMS Symposium in Pure Mathematics* 29 (1975): 165-185.

Hirsch, M. D., C. H. Papadimitriou, and S. A. Vavasis, "Exponential lower bounds for finding Brouwer fixed points," *Journal of Complexity*, 5 (1989): 379-416.

Huber, B., and B. Sturmfels, "A polyhedral method for solving sparse polynomial systems," mimeo (1993).

Kalai, E., and D. Samet, "Persistent equilibria," *International Journal of Game Theory*, 13 (1984): 129-144.

Kohlberg, E., and J. F. Mertens, "On the strategic stability of equilibria," *Econometrica*, 54 (1986): 1003-1037.

Kohlberg, E., and P. J. Reny, "An interpretation of consistent assessments," mimeo (1993).

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

Koller, D., and N. Megiddo, "Finding mixed strategies with small supports in extensive games," *International Journal of Game Theory*, (1994): forthcoming.

Koller, D., N. Megiddo, and B. von Stengel, "Efficient solutions of extensive two-person games," mimeo, Computer Science Division, University of California at Berkeley (1994).

Kreps, D. M., and R. Wilson, "Sequential equilibria," *Econometrica*, 50 (1982): 863-894.

Kuhn, H. W., "Extensive games and the problem of information," in *Contribution to the Theory of Games*, H. W. Kuhn and A. W. Tucker, eds., Vol II (1953): 193-216.

Kuhn, H. W., and J. G. MacKinnon, "Sandwich method for finding fixed points," *Journal of Optimization Theory and Applications*, 17 (1975): 189-204.

van der Laan, G., and A. J. J. Talman. "A restart algorithm for computing fixed points without an extra dimension," *Mathematical Programming*, 17 (1979): 74-84.

van der Laan, G., and A. J. J. Talman. "On the computation of fixed points in the product space of unit simplices and an application to noncooperative n-person games," *Mathematics of Operations Research*, 7 (1982): 1-13.

van der Laan, G., and A. J. J. Talman. "Simplicial approximation of solutions to the nonlinear complementarity problem with lower and upper bounds," *Mathematical Programming*, 38 (1987): 1-15.

van der Laan, G., A. J. J. Talman, and L. Van der Heyden. "Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling," *Mathematics of Operations Research*, 12 (1987): 377-397.

Lemke, C. E., "Bimatrix equilibrium points and mathematical programming," *Management Science*, 11 (1965): 681-689.

Lemke, C. E., and J. T. Howson, Jr., "Equilibrium Points in Bimatrix Games," *SIAM Journal on Applied Math*, 12 (1964): 413-423.

Mangasarian, O. L., "Equilibrium points in bimatrix games," *Journal of the Society for Industrial and Applied Mathematics*, 12 (1964): 778-780.

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

McKelvey, R. D., "A Liapunov function for Nash equilibria," mimeo, California Institute of Technology (1992).

McKelvey, R. D. and A. McLennan, "The maximal generic number of totally mixed Nash equilibria," **mimeo** , Department of Economics, University of Minnesota (1994).

McKelvey, R. D. and T. R. Palfrey, "An experimental analysis of the centipede game," *Econometrica*, 60 (1992): 803-836.

McKelvey, R. D. and T. R. Palfrey, "Quantal response equilibria for normal form games," Caltech Social Science Working Paper #883, (1993), forthcoming in *Games and Economic Behavior*

McKelvey, R. D. and T. R. Palfrey, "Quantal response equilibria for extensive form games," mimeo, (1994)

McLennan, A., "Justifiable beliefs in sequential equilibrium," *Econometrica*, 53 (1985): 889-904.

Merrill, O. H., "Applications and extensions of an algorithm that computes fixed points of certain upper semi-continuous point to set mappings," Ph. D. Thesis, University of Michigan, Ann Arbor, Michigan (1972).

Mertens, J., "Stable equilibria, a reformulation," mimeo, (1988).

Myerson, R. B., "Refinements of the Nash equilibrium concept," *International Journal of Game Theory*, 7 (1978): 73-80.

Murty, K. G., "Computational complexity and linear pivot methods," *Mathematical Programming Study*, 7 (1978): 61-73.

Nash, J. F., "Noncooperative games," *Annals of Mathematics*, 54 (1951): 289-295.

Rosen, J. B., "Existence and uniqueness of equilibrium points for concave $N-person$ games," *Econometrica*, 33 (1965): 520-534.

Rosenmüller, J., "On a generalization of the Lemke-Howson algorithm to noncooperative N-person games", *SIAM Journal of Applied Mathematics*, 1 (1971): 73-79.

Saigal, R., "On the convergence rate of algorithms for solving equations that are based on methods of complementary pivoting," *Mathematics of Operations Research*, 2 (1977): 108-124.

Scarf, H., "The approximation of fixed points of a continuous mapping," *SIAM Journal of Applied Mathematics*, 15 (1967): 1328-1343.

Scarf, H., *The Computation of Economic Equilibria* (New Haven: Yale University Press, 1973)

Schanuel, S. H., L. K. Simon, and W. R. Zame, "The algebraic geometry of games and the tracing procedure," in *Game Equilibrium Models, Vol II: Methods, Morals and Markets*, R. Selten, ed., (Berlin: Springer-Verlag, 1991).

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

Shapley, L., "A note on the Lemke-Howson algorithm," *Mathematical Programming Study*, 1 (1974): 175-189.

Shapley, L. S., "On balanced games without sidepayments," in *Mathematical Programming*, T. C. Hu, and S. M. Robinson, eds, (New York: Academic Press, 1973): 261-290.

Shapley, L., "On the accessibility of fixed points," in *Game Theory and Mathematical Economics* , O. Moeschlin, and D Pallaschke, eds. (North Holland, 1981): 367-377.

Smale, S., "A convergent process of price adjustment and global Newton methods," *Journal of Mathematical Economics*, 3 (1976): 107-120.

von Stengel, B., "LP representation and efficient computation of behavior strategies," mimeo (1994).

Talman, A. J. J., and Z. Yang, "A simplicial algorithm for computing proper equilibria of finite games, Center Discussion Paper 9418, Tilburg University, Tilburg (1994).

Tarski, A., *A Decision Method for Elementary Algebra and Geometry*, (Berkeley: University of California Press, 1951).

Todd, M. J., "On the computational complexity of piecewise-linear homotopy algorithms," *Mathematical Programming*, 24 (1982): 216-224.

Todd, M. J., *The Computation of Fixed Points and Applications*, (Berlin: Springer-Verlag, 1976).

van der Waerden, B. L., *Modern Algebra*, (New York: Ungar Publishing Co, 1949).

Wilson, R., "Computing equilibria of n-person games," *SIAM Journal of Applied Mathematics*, 21 (1971): 80-87.

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

Wilson, R., "Computing simply stable equilibria," *Econometrica*, 60 (1992): 1039-1070.

Yamamoto, Y., "A path-following procedure to find a proper equilibrium of finite games," *International Journal of Game Theory*, 22 (1993): 49-59.

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

Index

i-stopping simplex, 14

accessible, 12
actions, 20
adjacent, 10, 15
agents, 20
algorithms
  function minimization, 18
  Lemke-Howson, 5
  non-linear complementarity, 17
  simplicial subdivision, 13
    homotopy, 16
    restart, 16
  SLCP, 17
almost completely labeled, 14

basic solution, 6
basis, 6, 24
behavior strategy, 21
beliefs, 22
best response correspondence, 3

complementary, 7
completely labeled, 14
computational complexity, 13, 17, 36, 37, 44
consistent, 24, 29
consistent assessments, 22
cylindrical algebraic decomposition, 29

derangements, 38

endpoint, 11
equilibria
  finding all, 27
  number of, 37
  regular, 38
equilibrium refinements, 25
  perfect, 25, 28
  proper, 26
  sequential, 20, 22, 23, 26
  stable set, 26
Euclidean remainder sequence, 33
extensive form, 19
extraneous solution, 5

face, 14
facet, 14
feasible, 6
fixed point, 3

GAMBIT, 41
GCL, 41

homotopy method, 16

immediate predecessor, 20
immediately accessible, 12
index, 37
information partition, 20
initial assessment, 20
initial nodes, 20

Kronecker product, 34

labeling, 14
Lemke-Howson algorithm, 5
lex negative, 9
lex-feasible, 9
linear complementarity problem, 6
loop, 11

myopically rational, 23

Nash equilibrium, 3
nodes, 20
non-linear complementarity, 4, 17
nondegenerate, 9
noninitial nodes, 20
normal form, 2

ordered set, 14

path, 11

path following algorithms, 5
payoff, 22
payoff function, 2
perfect equilibrium, 25, 26, 28
pivot matrix, 7
players, 2
precedes, 20
proper equilibrium, 26
pure strategies, 2, 23

quadratfrei, 32
quadratfrei part, 32
quantified propositional formulas, 28

realization equivalent, 23
realization plan, 21
regular equilibria, 38
restart method, 16
resultant, 30
    sparse, 30
    u-resultant, 31

sample equilibrium, 1, 5
semi-algebraic set, 4, 28
sequence form, 21
sequential equilibrium, 20, 22, 23, 26
sign assignment, 29
simplex, 14
simplicial subdivision, 13
simplotope, 14
simply stable sets, 26
SLCP, 17
sparse resultant, 30
Sperner-proper, 14
stable set, 26
stationary point problem, 4, 18
strategic nodes, 20
strategy set, 2
support, 29

terminal nodes, 20
terminal simplex, 15
totally mixed, 22
triangulation, 14

u-resultant, 31
unbound, 28

zero sum games, 12