Computing the index of equilibrium components

Candidate Number: 32104

A Dissertation submitted to the Department of Mathematics
of the London School of Economics and Political Science
for the degree of Master of Science

August 31, 2016

Abstract

This dissertation provides proof, explanation and implementation of an algorithm described by Balthasar (2009) for computing the index of an equilibrium component of a degenerate bimatrix game. The natural definition of the index, due to Shapley (1974), for isolated equilibria of non-degenerate bimatrix games can be extended to connected equilibrium components of degenerate games. In principle, the index can be computed from a generic perturbation of the payoff matrices (to resolve degeneracy). Balthasar (2009) proposed a 'perturbation free' algorithm for index calculation. The algorithm simulates a lexicographic perturbation and provides an explicit method for index computation. It relies on a relationship between a set of bases corresponding to extreme equilibria that define an equilibrium component and the equilibria of a lexicographically perturbed game.

Contents

Chapter 1

Introduction

A bimatrix game is a two-player game in strategic form, one of the basic models of game theory. The game is specified by two matrices that define the payoffs to the two players when player 1 chooses a row and simultaneously player 2 chooses a column as his strategy. An equilibrium is a strategy pair such that no player can unilaterally improve his payoff. Nash (1951) introduced the concept of equilibrium and showed that an equilibrium exists if players can use mixed (that is, randomized) strategies. Given a bimatrix game, it is of interest to find all its equilibria. Avis et al. (2010) described state-of-the art algorithms that can find all equilibria of games of small dimensions (not more than 20 strategies per player).

This dissertation is concerned with a type of additional information about an equilibrium, called its *index*. The index has a natural definition if the equilibrium is isolated, which is the case when the game is non-degenerate (that is, there are no special linear relations between the payoffs). In that case, both players mix the same number of strategies, which defines a square sub-matrix of each payoff matrix. The index is then essentially the sign of the product of the determinants of these two payoff sub-matrices (see Definition 3.1).

In a degenerate game, there is no such straightforward method for index calculation. This is due to the fact that a degenerate game may have infinitely many equilibria, where in some of them players may mix between different numbers of strategies (which means that we cannot define the square sub-matrices mentioned earlier). However, the equilibria of a degenerate game can always be divided into connected components (see Proposition 2.14), and each component can be assigned an index. A general way to find the index of an equilibrium component is to perturb the game to make it non-degenerate, and then sum the indices (which must be ± 1) of the perturbed equilibria near the original equilibrium component. This dissertation contains the proof, explanation and implementation of an algorithm proposed by Balthasar (2009) to calculate the index of an equilibrium component. The algorithm simulates a so-called 'lexicographic' perturbation, which makes it easy to identify and calculate the indices of equilibria that are close to the given component. Summing the indices of these equilibria then yields the index of the equilibrium component.

Chapter 2

Bimatrix games

2.1 Preliminaries

A bimatrix game is a two-player game specified by two real payoff matrices A and B, which have the same dimensions $m \times n$ for some positive integers m and n. The m rows correspond to the pure strategies of player 1, who has strategy set $M = \{1, \ldots, m\}$, and the n columns correspond to the pure strategies of player 2, who has strategy set $N = \{1, \ldots, n\}$. The players choose their strategies simultaneously and independently and receive their payoffs. When player 1 chooses to play strategy $i \in M$ and player 2 chooses to play strategy $j \in N$, the payoffs will be a_{ij} for player 1 and b_{ij} for player 2, where a_{ij} and b_{ij} denote the entries of A and B, respectively.
Each player is interested in maximizing his own payoff. Players are allowed to mix (randomize) between their pure strategies and thus play a strategy called a *mixed strategy*. A mixed strategy for a player is defined by a probability distribution over the set of pure strategies of that player. Specifically, a mixed strategy x for player 1 is an m-component vector $(x_1, \ldots, x_m)$ such that $x_i \geq 0$ for every $i \in M$ and $\sum_{i=1}^{m} x_i = 1$. A mixed strategy $y \in \mathbb{R}^n$ for player 2 is defined analogously. Players are then interested in maximizing their expected payoffs. Choosing to play a pure strategy i is basically playing a mixed strategy where $x_i = 1$, so we can treat all strategies as mixed. A strategy x of a player is said to be the *best response* to a strategy y (of his opponent) if it maximizes his expected payoff (that is, if changing the strategy unilaterally will not improve his payoff). If x and y are best responses to each other, then (x, y) is called an *equilibrium*. One of the fundamental theorems of game theory, proved by Nash (1951), states that every strategic-form game has at least one equilibrium in mixed strategies.

2.2 Review of linear programming

Finding a best response against a fixed opponent strategy is a linear programming problem. The aim of linear programming is to find an optimal solution (or to prove that one does not exist) to a linear function, called the *objective function*, subject to a set of *linear constraints*.

Linear programs in standard equality form

Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The optimization (maximization) linear program in standard equality form is defined as

$$\begin{aligned} \text{maximize} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq \mathbf{0} \end{aligned} \tag{2.1}$$

where $\mathbf{0}$ denotes the zero vector with n components and x is the n-vector $(x_1, \ldots, x_n)^\top$ of variables. The expression $c^\top x$ is the objective function, A and b determine m linear constraints, and $x \geq \mathbf{0}$ defines an additional n non-negativity constraints.
Now we define some terms that will be useful for the rest of this dissertation.

Definition 2.1 (Feasible and optimal solutions)**.**

- **Feasible solution** A point x is a feasible solution for the optimization problem (2.1) if it satisfies all the constraints.
- **Feasible region** The feasible region for the optimization problem is the set of all feasible solutions.
- **Optimal solution** An optimal solution for a maximization problem is a feasible solution x^* that satisfies $c^\top x^* \geq c^\top x$ for every feasible solution x.

Definition 2.2 (Basis)**.** A set $\beta \subseteq \{1, \ldots, n\}$ is said to be a *basis* of $A \in \mathbb{R}^{m\times n}$ if

(i) $|\beta| = m$;

(ii) the matrix A restricted to the columns indexed by β, denoted by A_β, is non-singular.

Definition 2.3 (Basic solution)**.** A point $x \in \mathbb{R}^n$ is a *basic solution* of (2.1) if

(i) $Ax = b$;

(ii) there exists a basis β of A such that $x_j = 0$ for every $j \notin \beta$.

If x is basic and feasible, we say that x is a *basic feasible solution* and that β is a *feasible basis*. We will denote the complement of β, i.e. $\{1, \ldots, n\} \setminus \beta$, by $\mathcal{N}$, the variables of x that correspond to β, called the *basic variables*, by x_β, and the non-basic variables by $x_\mathcal{N}$. Given a basis β, the corresponding unique basic solution will be

$$\begin{aligned} x_\beta &= A_\beta^{-1} b \\ x_\mathcal{N} &= \mathbf{0} \end{aligned}$$

2.3 Bimatrix game as a linear complementarity problem

Given a strategy y of player 2, player 1 seeks to find a strategy x, out of all possible mixed strategies, that maximizes the expression $x^\top Ay$ (player 1's expected payoff). This can be formulated as the following linear program:

$$\begin{aligned} \text{maximize} \quad & x^\top (Ay) \\ \text{subject to} \quad & x^\top \mathbf{1}_m = 1 \\ & x \geq \mathbf{0} \end{aligned} \tag{2.2}$$

where $\mathbf{1}_k$ denotes the k-component vector with all components equal to 1. Analogously, player 2 seeks the optimal response to a given opponent strategy x, which can be formulated as the linear program

$$\begin{aligned} \text{maximize} \quad & y^\top (B^\top x) \\ \text{subject to} \quad & y^\top \mathbf{1}_n = 1 \\ & y \geq \mathbf{0} \end{aligned} \tag{2.3}$$

The following well-known proposition asserts that a strategy x is the best response to strategy y if and only if all pure strategies played with positive probability are pure best responses to y.

Proposition 2.4 (Best response condition, Nash 1951)**.** *Let x and y be mixed strategies of players 1 and 2, respectively. Then x is the best response to y if and only if for all $i \in M$,*

$$x_i > 0 \implies (Ay)_i = u = \max_{k \in M} \big\{(Ay)_k\big\};$$

similarly, y is the best response to x if and only if for all $j \in N$,

$$y_j > 0 \implies (B^\top x)_j = v = \max_{k \in N} \big\{(B^\top x)_k\big\}.$$

The above is a modified 'two-player version' of the original proposition of Nash (1951, pg. 287), which pertains to a general n-player strategic-form game. All pure strategies played with positive probability must have maximum, and hence equal, expected payoff to that player. This implies that the vectors $x^\top$ and $(\mathbf{1}_m u - Ay)$, which are both non-negative, should be complements of each other; that is, if one has a positive component, the other should take value zero in that component, and vice versa. Similarly, $y^\top$ and $(\mathbf{1}_n v - B^\top x)$ should be complementary as well. This can be summed up in the next theorem (von Stengel, 1996, Theorem 2.4).

Theorem 2.5. *In an $m \times n$ bimatrix game (A, B), (x, y) is an equilibrium if and only if there exist $u, v \in \mathbb{R}$, $r \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ that satisfy*

$$\begin{aligned} y^\top \mathbf{1}_n &= 1 \\ -\mathbf{1}_m u + Ay + r &= \mathbf{0} \\ y, r &\geq \mathbf{0} \end{aligned} \tag{2.4}$$

and

$$\begin{aligned} x^\top \mathbf{1}_m &= 1 \\ -\mathbf{1}_n v + B^\top x + s &= \mathbf{0} \\ x, s &\geq \mathbf{0} \end{aligned} \tag{2.5}$$

such that

$$\begin{aligned} x^\top r &= 0 \\ y^\top s &= 0. \end{aligned} \tag{2.6}$$

This problem is called *linear complementarity problem* (abbreviated LCP). It consists of a set of linear constraints and a complementarity condition. The linear constraint systems (2.4) and (2.5) have variables u and v that are free (i.e. can take any value), and x, y, r and s are constrained to be non-negative. The vectors r and s consist of slack variables and are generally discarded after

(u, y) and (v, x) are found. System (2.6) is called the complementarity condition; it states that the two non-negative vectors in each of the pairs (x, r) and (y, s) should complement each other, so that if one has a positive component then the other is zero in the corresponding component. We require the free variables u and v to always be basic. To avoid special treatment of them, we can add a positive integer to all entries of the payoff matrices to make all payoffs positive, without changing the structure of the game. This will make u and v basic and positive in every basic feasible solution. Therefore, from now on we will always assume that A and B are positive. It will be convenient to view a feasible solution to this LCP as two separate feasible solutions (u, y) and (v, x) (to (2.4) and (2.5), respectively) that satisfy the complementarity condition. The vectors x and y will then define the equilibrium mixed strategies, while u and v will represent the equilibrium payoffs.
For a mixed strategy x, define the support of x, denoted by $\text{supp}(x)$, to be the set of pure strategies that are played with positive probability.

Definition 2.6. Let $x \in \mathbb{R}^k$ be a mixed strategy; then

$$\text{supp}(x) = \{i \mid x_i > 0,\ 1 \le i \le k\}.$$

We will denote the size of the support of a strategy x by $|\text{supp}(x)|$.

Definition 2.7. A game is said to be *non-degenerate* if every mixed strategy x of player 1 has at most $|\text{supp}(x)|$ pure best responses, and the same holds for every mixed strategy y for player 2. Otherwise, the game is said to be *degenerate*.

An equivalent condition for non-degeneracy, drawn from von Stengel (1996, Theorem 2.7), is formulated in the following proposition.

Proposition 2.8. *Let (A, B) be a bimatrix game with $A, B > 0$. The game is non-degenerate if in every basic feasible solution to (2.4) and (2.5), all basic variables have positive values.*

Example 2.9. Consider the 2×3 degenerate bimatrix game shown below (the boxes indicate best-response payoffs).

I \ II	l	c	r
T	**2**, 0	1, **4**	0, **3**
B	0, **2**	1, 2	**2**, 1

This game is degenerate since mixed strategy $(\frac{1}{2}, \frac{1}{2})$ of player 1, which has support size 2, has three pure best responses: l, c and r.

An important and well-known property of non-degenerate games, which can be immediately deduced from the best response condition (Proposition 2.4), is that in equilibrium both players mix between the same number of pure strategies (i.e. have equal support size). This is formulated in the following proposition.

Proposition 2.10. *If (x, y) is an equilibrium of a non-degenerate bimatrix game (A, B), then x and y have equal support size.*

Proof. Let (x, y) be an equilibrium of the game (A, B). The best response condition states that any pure strategy in the support of x should be the best response to y. Since (A, B) is non-degenerate, y has at most $|\text{supp}(y)|$ pure best responses. Therefore, $|\text{supp}(x)| \leq |\text{supp}(y)|$. Similarly, $|\text{supp}(y)| \leq |\text{supp}(x)|$, and hence $|\text{supp}(x)| = |\text{supp}(y)|$. □

Therefore, in non-degenerate games, only pairs of *basic* feasible solutions (to (2.4) and (2.5)) are candidates to fulfil the complementarity condition. Since the number of possible basic feasible solutions for each player, and hence the number of pairs of basic feasible solutions, is finite, the number of equilibria that we can expect to find is finite as well. However, this is not true for degenerate games, which could have an infinite number of equilibria. Nevertheless, the set of equilibria of a degenerate game can be partitioned into finitely many connected equilibrium components defined by so-called *extreme* equilibria.

Definition 2.11. An equilibrium is said to be *extreme* if it cannot be written as a convex combination of other equilibria.

It is easy to see that the definitions of equilibrium and extreme equilibrium coincide for non-degenerate games, in which the equilibria are therefore said to be 'isolated'. The feasible regions of the linear constraint systems (2.4) and (2.5) define convex sets. The following more general definition of extreme feasible solution, taken from (Dantzig, 1963, pg. 154), is an equivalent form of Definition 2.11, when considering the feasible regions of (2.4) and (2.5) as convex sets.

Definition 2.12. Any point x in a convex set C which is not a midpoint of the line segment joining two other points in C is said to be *extreme* or is called a *vertex* of the convex set.

We will prove the intuitive link between extreme equilibria and basic feasible solutions inspired by the proof in Dantzig (1963, pg. 154, Theorem 3), using Definition 2.12.

Theorem 2.13. *Any feasible solution* (u, y) *to the linear constraints* (2.4) *is basic if and only if it is an extreme solution.*

Proof. As mentioned earlier, we assume $A > 0$, which makes u positive and basic in every basic feasible solution. Let $z = (u, y)$ be a basic feasible solution for (2.4) and let β be a corresponding basis. Suppose that z is the midpoint between two other feasible points z' and z''. Now $z_i = 0$ for all $i \notin \beta$ (as non-basic variables take the value zero), which implies that $\frac{1}{2}(z'_i + z''_i) = 0$ for all $i \notin \beta$. But $z'_i \geq 0$ and $z''_i \geq 0$ since z' and z'' are feasible. This is possible if and only if $z'_i = z''_i$ for all $i \notin \beta$, but this uniquely determines the values of z'_i and z''_i for all $i \in \beta$, which implies that $z' = z'' = z$. Hence (u, y) is extreme.
For the converse, let $z = (u, y)$ be a non-basic feasible solution. We need to show that z is not extreme. The linear constraint system (2.4) is in the form of $Cz = q$, for a matrix C, vector of variables z and right-hand side vector q. Let P be the set of all indices of positive components of z and let Q be its complement. Since z is non-basic, there is linear dependence between the columns of the sub-matrix C_P (the matrix C restricted to columns indexed by P). Therefore, we can find $z'_P \neq 0$ such that $C_P z'_P = 0$. Define the components z'_i of a vector z' to equal to z'_P for all $i \in P$ and zero for all $i \in Q$. Consider the two vectors $z + \epsilon z'$ and $z - \epsilon z'$. For small enough ϵ, those two vectors are both feasible; so we have

$$z = \frac{1}{2}(z + \epsilon z') + \frac{1}{2}(z - \epsilon z'),$$

and therefore z is not extreme. □

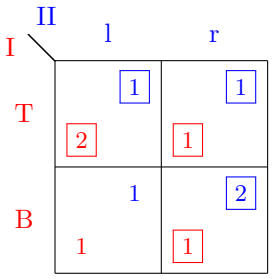
Theorem 2.13 holds for (v, x) and (2.5) as well, with an analogous proof.
The following proposition from Avis et al. (2010, Proposition 4), which we state in a slightly modified form to fit our purposes, shows how the set of all equilibria can be divided into connected components.

Proposition 2.14. *Let (x, y) be an equilibrium of a bimatrix game (A, B), and let u and v be the corresponding equilibrium payoffs. Then $(v, x) \in \text{conv}(V)$ and $(u, y) \in \text{conv}(U)$ for some convex feasible sets U and V for (2.4) and (2.5), respectively, and every pair $(u, v) \in U \times V$ satisfies the complementarity condition (2.6).*

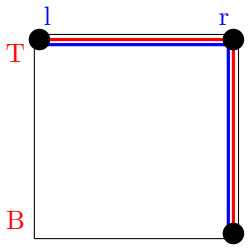
As explained in Avis et al. (2010), Proposition 2.14 tells us that all equilibria of a bimatrix game can be described by finitely many pairs of vertices of the convex feasible regions of (2.4) and (2.5). Consider the bipartite graph whose two sets of vertices are the two sets of basic feasible solutions to (2.4) and (2.5), where two vertices are joined by an edge if they satisfy the complementarity condition (i.e. form an equilibrium). The maximal cliques (maximal complete bipartite graphs) are the sets $U \times V$ which define the equilibria sets $\text{conv}(U) \times \text{conv}(V)$ in Proposition 2.14. These sets are called *maximal Nash subsets*; union of all of them describe all equilibria of a given bimatrix game. Maximal Nash subsets may be non-disjoint, as shown in Example 2.15 below. The inclusion-wise maxima of all maximal Nash subsets yield the *connected equilibrium components* of the game.

Example 2.15. Consider the degenerate bimatrix game shown in Figure 2.1.

Figure 2.1: Equilibrium component



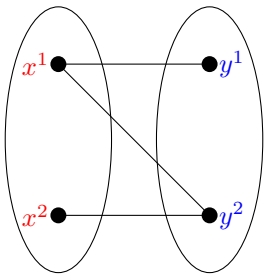
Matrix layout of a degenerate bimatrix game



Best response correspondence diagram; black circles represent extreme equilibria.

The best response correspondence diagram on the right of Figure 2.1 can be read as follows. The vertical axis of the square corresponds to the mixed strategy set of player 1; similarly, the horizontal axis corresponds to the mixed strategy set of player 2. The red and blue lines represent the best responses of players 1 and 2, respectively, as functions of their opponent's strategy. Wherever the two lines overlap or touch, the corresponding strategy pair is an equilibrium. Extreme equilibria are marked with black circles.
This game is clearly degenerate. As we can see from the best response correspondence diagram, it has a single equilibrium component. Denote the basic feasible solutions that correspond to the pure strategies T, B, l and r by x^1, x^2, y^1 and y^2, respectively. The corresponding bipartite graph is displayed in Figure 2.2.

Figure 2.2: Bipartite graph of the equilibria set for Example 2.15



The two maximal cliques (maximal Nash subsets) are (x^1, y^1, y^2) and (x^1, x^2, y^2). Since they share the equilibrium (i.e. edge) (x^1, y^2), they form a single connected component. In particular, the collection of all equilibria can be described as the union of the two sets

$$\{(1,0),\ (1-q,q) \mid\ q \in [0,1]\}$$

and

$$\{(1-p,p),\ (0,1) \mid\ p \in [0,1]\}.$$

Chapter 3

The index of an equilibrium

3.1 Introduction

Shapley (1974) introduced the notion of *index* of an equilibrium. The index is an integer which is assigned to each equilibrium and is related to concepts of strategic stability (see, for example, von Schemde and von Stengel (2008)). In non-degenerate games, strategies in equilibrium have equal support size (Proposition 2.10). In that case, the sub-matrices restricted to the rows and columns in the support of the equilibrium strategies are square and non-singular, and the index is defined as follows.

Definition 3.1 (Shapley 1974)**.** Let (x, y) be an equilibrium of a non-degenerate bimatrix game (A, B) with positive payoff matrices, and let A_{xy} and B_{xy} be the corresponding sub-matrices restricted to the rows in the support of x and columns in the support of y. Then the index of (x, y) is defined as

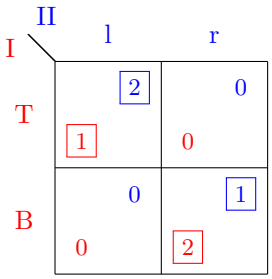
$$(-1)^{|\mathrm{supp}(x)|+1} \operatorname{sign}\big(\det(A_{xy}) \cdot \det(B_{xy})\big).$$

The index has several properties that arise immediately from Definition 3.1. Non-degeneracy implies that the corresponding sub-matrices A_{xy} and B_{xy} of an equilibrium (x, y) are non-singular and therefore the index is restricted to the values ± 1. Moreover, it is easy to see that any pure strategy equilibrium will have index $+1$ (assuming $A, B > 0$). The *Lemke–Howson algorithm*, a complementary pivoting algorithm for non-degenerate bimatrix games, can divide the set of all equilibria into disjoint pairs through paths called Lemke–Howson paths (including a special 'artificial' equilibrium of empty support); see Lemke and Howson (1964) for more details. An important property of the index, proved by Shapley (1974), is that equilibria at different endpoints of a Lemke–Howson path have opposite indices. Since the artificial equilibrium has index -1 by definition, we can deduce that the sum of all indices should be $+1$. See von Schemde and von Stengel (2008) for more details about the index of an equilibrium in non-degenerate games.

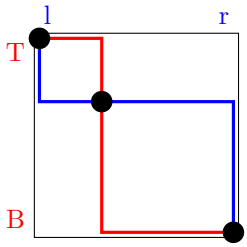
Example 3.2. Consider the bimatrix game depicted in Figure 3.1, known as 'Battle of the Sexes'. Find the indices of its equilibria.

This non-degenerate bimatrix game has three equilibria, two pure and one mixed. Each of the two pure equilibria, (T, r) and (B, l), has support of size 1, and the sign of the corresponding determinant is positive (as each sub-matrix is 1×1 with a single positive entry, namely the payoff to the player). Hence, these pure equilibria have index $+1$. However, the mixed equilibrium has

Figure 3.1: Bimatrix game 'Battle of the Sexes'



Matrix layout; the pure best response payoffs are enclosed in squares.



Best response correspondence diagram; extreme equilibria are marked with black circles.

support of size 2. Its index is calculated as

$$(-1)^{2+1}\,\mathrm{sign}\left(\begin{vmatrix}1 & 0\\ 0 & 2\end{vmatrix}\cdot\begin{vmatrix}2 & 0\\ 0 & 1\end{vmatrix}\right) = (-1)\,\mathrm{sign}(2\cdot 2) = -1\cdot 1 = -1.$$

Thus we can verify that the sum of all indices is indeed $+1$.

For degenerate games, however, we do not have such explicit formula. The sub-matrices in Definition 3.1 may not be square, so their determinants may be undefined. Moreover, a degenerate game can have an infinite number of equilibria. The notion of index of an equilibrium can be extended to degenerate games, but explaining it involves some advanced topological concepts. We will not elaborate on this issue here; for more details see Balthasar (2009). Recall that the equilibria of a degenerate game can be divided into finitely many connected equilibrium components. Therefore, in a degenerate game, we will assign an index to each equilibrium component, but this index will no longer be constrained to the values $\{+1, -1\}$ and can take any integer value. Nonetheless, many other properties of the index still hold. In particular, the indices of all equilibrium components should still add up to $+1$. A general method for calculating the index of an equilibrium component is to choose a non-degenerate perturbation (i.e. make a slight modification to the payoff matrices to resolve degeneracy), calculate the indices of the equilibria in the perturbed game that are 'close' to the given component in the original game (using Definition 3.1) and then add up these indices.

3.2 Lexicographic perturbation

Degenerate games can be solved by a lexicographic method that makes them non-degenerate. In the following section we will describe how to perturb a game lexicographically and analyse the relationship between the equilibria of the original and the perturbed games. This section follows Balthasar (2009, sec. 2.3); we fill up omitted parts in some of the proofs and provide extra examples along the way.

For an $m \times n$ bimatrix game (A, B) and $\epsilon > 0$, we define the $m \times n$ lexicographically perturbed game $(\mathcal{A}, \mathcal{B})$ by

$$\mathcal{A} = A - \mathcal{E}_1 \qquad \text{and} \qquad \mathcal{B} = B - \mathcal{E}_2$$

where $\mathcal{E}_1$ and $\mathcal{E}_2$ are $m \times n$ matrices given by

$$\mathcal{E}_1 = \begin{bmatrix} \epsilon & \cdots & \epsilon \\ \epsilon^2 & \cdots & \epsilon^2 \\ \vdots & \ddots & \vdots \\ \epsilon^m & \cdots & \epsilon^m \end{bmatrix}, \qquad \mathcal{E}_2 = \begin{bmatrix} \epsilon & \epsilon^2 & \cdots & \epsilon^n \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon & \epsilon^2 & \cdots & \epsilon^n \end{bmatrix}.$$

Applying Theorem 2.5 to $(\mathcal{A}, \mathcal{B})$, we get that its equilibria are given by the solutions to the following systems.

$$\begin{aligned} \mathbf{1}_n^\top y &= 1 \\ -\mathbf{1}_m^\top u + \mathcal{A} y + r &= \mathbf{0} \\ y, r &\geq \mathbf{0} \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} \mathbf{1}_m^\top x &= 1 \\ -\mathbf{1}_n^\top v + \mathcal{B}^\top x + s &= \mathbf{0} \\ x, s &\geq \mathbf{0} \end{aligned} \tag{3.2}$$

such that

$$\begin{aligned} x^\top r &= 0 \\ y^\top s &= 0 \end{aligned} \tag{3.3}$$

Since $\mathbf{1}^\top y = 1$, we have

$$\mathcal{A} y = (A - \mathcal{E}_1) y = Ay - \mathcal{E}_1 y = Ay - \begin{bmatrix} \epsilon & \ldots & \epsilon^m \end{bmatrix}^\top.$$

Similarly, since $\mathbf{1}^\top x = 1$, we obtain

$$\mathcal{B}^\top x = B^\top x - \begin{bmatrix} \epsilon & \ldots & \epsilon^n \end{bmatrix}^\top.$$

Therefore, we find that the systems (3.1)–(3.3) are equivalent to

$$\begin{aligned} \mathbf{1}_n^\top y &= 1 \\ -\mathbf{1}_m^\top u + Ay + r &= \begin{bmatrix} \epsilon & \cdots & \epsilon^m \end{bmatrix}^\top \\ y, r &\geq \mathbf{0} \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \mathbf{1}_m^\top x &= 1 \\ -\mathbf{1}_n v + B^\top x + s &= \begin{bmatrix} \epsilon & \cdots & \epsilon^n \end{bmatrix}^\top \\ x, s &\geq 0 \end{aligned} \tag{3.5}$$

such that

$$\begin{aligned} x^\top r &= 0 \\ y^\top s &= 0 \end{aligned} \tag{3.6}$$

Proposition 3.3. *There exists a positive number ϵ_0 such that for all $0 < \epsilon < \epsilon_0$, the perturbed bimatrix game $(\mathcal{A}, \mathcal{B})$ is non-degenerate.*

Before proceeding with the proof, we need to define some notation to be used in the rest of this section. The system of linear constraints (3.4) is in the form $Cz = q$ for a $(1 + m) \times (1 + n + m)$ matrix C, a $(1 + n + m)$-component vector of variables z and an $(1 + m)$-component right-hand side vector q. Given a basis β for C, its corresponding basic matrix will be denoted by C_β. Similar notation can be defined for (3.5), though it is not needed at this point.

Proof of Proposition 3.3. According to Theorem 2.8, in order to prove non-degeneracy we need to show that in every basic feasible solution for (3.4) and (3.5), all basic variables are positive. We will prove this for (3.4); an analogous argument will hold for (3.5). Let β be a basis for C; then the corresponding basic variables vector $z_\beta \in \mathbb{R}^{m+1}$ is given by

$$z_\beta = C_\beta^{-1} q = C_\beta^{-1} \cdot \begin{pmatrix} 1 & \epsilon & \dots & \epsilon^m \end{pmatrix}^\top .$$

We need to show that $z_\beta > \mathbf{0}$. Let us denote the entries of C_β^{-1} by $\bar{c}_{i,j}$. We then obtain that the ith component of z_β is

$$(z_\beta)_i = \bar{c}_{i,1} + \epsilon \cdot \bar{c}_{i,2} + \dots + \epsilon^m \cdot \bar{c}_{i,m+1}.$$

Observe that for sufficiently small ϵ, columns with lower index have more 'weight' and will determine the sign of $(z_\beta)_i$. Thus, for row i, the first column with a non-zero entry will set the sign of $(z_\beta)_i$. This property of a row of C_β is referred to as *lexico-positivity*. Moreover, since C_β^{-1} has full rank, it does not contain a zero row. This implies that z_β will not have a zero entry. Hence, if β is a feasible basis, i.e. if its solution satisfies all the non-negativity constraints, then z_β will be positive. □

Proposition 3.4. *A basis β will be feasible for* (3.4) *or* (3.5) *if and only if its corresponding basic matrix is lexico-positive.*

Proof. We have already established this in the proof of Proposition 3.3. □

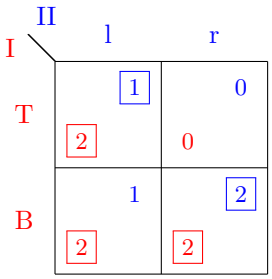
A matrix is said to be lexico-positive if all its rows are lexico-positive, that is, the first non-zero entry is positive.

Remark 3.5. *For a lexicographic perturbation, an ϵ is considered 'sufficiently small' if it satisfies the conditions of Proposition 3.3.*

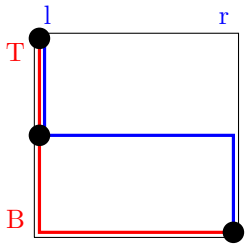
Now that we have set up a non-degenerate perturbation of the original game, we need to analyse the relationship between the equilibria of the original and perturbed games. In particular, for each equilibrium component of the original game, we need to determine which equilibria of the perturbed game are close to it. As proved in Theorem 2.13, each extreme equilibrium has a corresponding basic feasible solution pair to (2.4) and (2.5). Given a basis, the corresponding basic solution is unique. However, in degenerate cases, basic variables may have value zero and therefore several bases may give rise to the same basic solution. In other words, an extreme equilibrium could be associated with several different pairs of bases.

Example 3.6. The bimatrix game in Figure 3.2 demonstrates how an extreme equilibrium can be associated with two different pairs of bases. The game is degenerate because of the payoff matrix A to player 1. The pure strategy l has two pure best responses, T and B. The pure equilibrium

Figure 3.2: Bimatrix game for Example 3.6



Matrix layout; the pure best response payoffs are enclosed in squares.



Best response correspondence diagram; extreme equilibria are marked with black circles.

(T, l) can be associated with the following two pairs of feasible bases: $(u, y_1, r_1), (v, x_1, s_2)$ and $(u, y_1, r_2), (v, x_1, s_2)$.

Notice that the matrices on the left-hand sides of the perturbed ((3.4) and (3.5)) and unperturbed ((2.4) and (2.5)) systems are the same; the only difference between the perturbed and unperturbed systems is in the right-hand sides. This means that the two systems have the same set of bases. One relationship between the feasibility of the two systems is described in the following proposition.

Proposition 3.7. *A feasible basis β for the perturbed system* (3.4) *(or* (3.5)*) will be feasible for the unperturbed system* (2.4) *(or* (2.5)*).*

Proof. We will prove this result for (3.4) and (2.4); the proof for (3.5) and (2.5) is analogous. The basic variable vector z_β for the perturbed system is

$$z_\beta = C_\beta^{-1} q = C_\beta^{-1} \cdot \begin{pmatrix} 1 & \epsilon & \dots & \epsilon^m \end{pmatrix}^\top.$$

To get the corresponding basic variable for the original system (2.4), we can plug in $\epsilon = 0$; that is, z_β is equal to the first column of C_β^{-1}. This column does contain a negative entry, or else β would not have been feasible for (3.4). Therefore, β is feasible for (2.4) as well. □

So, how does this relate to the equilibria of the original and perturbed games? Theorem 2.5 states that an equilibrium corresponds to a pair of feasible solutions (to (2.4) and (2.5)) that satisfy the complementarity condition (2.6). Proposition 3.4 asserts that a basis of the original system will be feasible for the corresponding perturbed system if and only if its basic matrix is lexico-positive. We need to determine which of these lexico-feasible pairs of bases will satisfy the complementarity condition. For this purpose we make the following definition.

Definition 3.8 (Complementary bases)**.** A pair of bases (α, β) for the linear systems (2.4) and (2.5) is said to be *complementary* if

$$y_i \in \alpha \implies s_i \notin \beta$$

and

$$x_i \in \beta \implies r_i \notin \alpha.$$

In other words, if a variable that corresponds to a pure strategy is basic, then its corresponding slack variable ought to be non-basic. Notice that since basic variables can take the value zero

(in degenerate cases), a pair of feasible bases being complementary is a stronger condition than their satisfying the complementarity condition (2.6); that is, being complementary implies that the complementarity condition holds, but not the other way around. For non-degenerate games, however, the two conditions coincide. Now we can assemble all the pieces to establish a proposition which summarizes the relationship between the bases of the original game and the equilibria of the lexicographically perturbed game. This relationship underlies the algorithm we will present in Chapter 4.

Proposition 3.9. *The strategy pair (x, y) is an equilibrium of the perturbed game $(\mathcal{A}, \mathcal{B})$ if and only if its corresponding pair of bases (α, β) is lexico-feasible and complementary for the original systems* (2.4) *and* (2.5).

By 'lexico-feasible and complementary' we mean that both bases are feasible, their corresponding basic matrices are lexico-positive and they are complementary bases.

Proof. Let (α, β) be a pair of lexico-feasible and complementary bases for (2.4) and (2.5). By Proposition 3.4, the corresponding basic solutions for (3.4) and (3.5) are feasible. Furthermore, the bases being complementary implies that the complementarity condition (2.6) holds, and hence (α, β) corresponds to an equilibrium of the perturbed game $(\mathcal{A}, \mathcal{B})$.
For the converse, let (x, y) be an equilibrium of $(\mathcal{A}, \mathcal{B})$. Non-degeneracy of $(\mathcal{A}, \mathcal{B})$ implies that (x, y) has a unique pair of bases (α, β) for the corresponding perturbed systems (3.4) and (3.5) with which it is associated. We need to show that (α, β) is lexico-positive and complementary for (2.4) and (2.5). It follows from Proposition 3.7 that (α, β) is feasible for (2.4) and (2.5). In addition, Proposition 3.4 gives that (α, β) is lexico-positive. Finally, since (x, y) is an equilibrium, it satisfies the complementarity condition. Non-degeneracy of $(\mathcal{A}, \mathcal{B})$ implies that (α, β) is also complementary. Consequently, (α, β) is lexico-feasible and complementary. □

3.3 The lex-index

As stated in the introduction, the index of an equilibrium component will equal the sum of the indices of all equilibria that are close to it in a perturbed version of the game. Proposition 2.14 tells us how equilibrium components are specified by (one or more, but finitely many) extreme equilibria. Because of degeneracy, each of these extreme equilibria could be associated with more than one pair of bases (see Example 3.6). In the previous section we introduced and proved the link between a subset of these pairs of bases, namely the lexico-feasible and complementary pairs, and those equilibria of the non-degenerate lexicographically perturbed game that are 'close' to the original extreme equilibria. We seek to calculate the indices of these 'nearby' equilibria and add them up. Balthasar (2009) introduced the concept of *lex-index*, an integer that is assigned to each extreme equilibrium and which measures its contribution to the index of the component. We will show that the lex-index has the desired property that the sum of the lex-indices of all the extreme equilibria specifying an equilibrium component yields the index of that component.
This section follows Balthasar (2009, sec. 2.4).
Let us start with some notation we will need in this section. For a square $k \times k$ matrix M, denote by $[M|_i \, \mathbf{1}]$ the matrix obtained from M by replacing the ith column with the vector $\mathbf{1}_k$. In addition, define the vector σ_M as follows.

$$\sigma_M = \begin{bmatrix} \det(M) & -\det([M|_1 \, \mathbf{1}]) & \dots & -\det([M|_k \, \mathbf{1}]) \end{bmatrix}$$

Recall that a non-zero vector is said to be *lexico-positive* if its first non-zero entry is positive; otherwise we will say that it is *lexico-negative*.
Define $\zeta(M)$, the 'sign of M', by

$$\zeta(M) = \begin{cases} +1 & \text{if } \sigma_M \text{ is lexico-positive} \\ -1 & \text{if } \sigma_M \text{ is lexico-negative} \\ 0 & \text{if } \sigma_M = 0 \end{cases}$$

The following theorem introduces a method to calculate the index of an equilibrium of the lexicographically perturbed game, without performing an actual perturbation. This method is the key ingredient of the algorithm we will present in the next chapter. Together with the unique link between the equilibrium of the perturbed game and an extreme equilibrium of the original game through its pair of bases (see Proposition 3.9) we finalise the recipe for index calculation.

Theorem 3.10. *Let (A, B) be a bimatrix game and $(\mathcal{A}, \mathcal{B})$ its corresponding lexicographically perturbed game. Let (x, y) be an equilibrium of $(\mathcal{A}, \mathcal{B})$, let (α, β) be the corresponding lexico-feasible and complementary pair of bases (unique by Proposition 3.9), let t be the number of variables x_i that are basic (which is equal to the number of basic variables y_j due to complementarity), and let $(A_{\alpha\beta}, B_{\alpha\beta})$ be the square sub-matrices of (A, B) restricted to rows and columns of basic variables of α and β, respectively. Then the index of (x, y) is*

$$(-1)^{t+1} \zeta(A_{\alpha\beta}^{\top})\, \zeta(B_{\alpha\beta}) \tag{3.7}$$

We start by proving the following auxiliary lemma.

Lemma 3.11. *Let M be a $k \times k$ matrix. For any vector $\boldsymbol{v} \in \mathbb{R}^k$, let C be the $k \times k$ matrix*

$$C = \begin{bmatrix} \boldsymbol{v} & \dots & \boldsymbol{v} \end{bmatrix}.$$

Then

$$\det(M + C) = \det(M) + \sum_{i=1}^{k} \det([M|_i\, \boldsymbol{v}]).$$

Proof. Recall that the determinant function is multi-linear; that is, for a $k \times k$ matrix A with columns $a_1, \dots, a_k$ and a vector $\boldsymbol{v} \in \mathbb{R}^k$, we have

$$\begin{aligned} &\det\left(\begin{bmatrix} a_1 & \dots & a_i + \boldsymbol{v} & \dots & a_k \end{bmatrix}\right) \\ &\quad = \det\left(\begin{bmatrix} a_1 & \dots & a_i & \dots & a_k \end{bmatrix}\right) + \det\left(\begin{bmatrix} a_1 & \dots & \boldsymbol{v} & \dots & a_k \end{bmatrix}\right) \\ &\quad = \det(A) + \det(A|_i\, v) \end{aligned}$$

By applying the above to $\det(M + C)$, while eliminating all determinants of matrices with two identical columns (since the determinant would be zero), we are left with

$$\det(M + C) = \det(M) + \sum_{i=1}^{k} \det([M|_i\, \boldsymbol{v}])$$

as claimed. □

Now we are ready to prove Theorem 3.10.

Proof of Theorem 3.10. The game $(\mathcal{A}, \mathcal{B})$ is non-degenerate (see Proposition 3.3). Therefore, by Definition 3.1, the index of (x, y) is

$$(-1)^{t+1}\,\mathrm{sign}\big(\det(\mathcal{A}_{\alpha\beta})\cdot\det(\mathcal{B}_{\alpha\beta})\big).$$

Let us calculate the sign of $\det(\mathcal{A}_{\alpha\beta})$. Denote the support of x by $\{i_1,\ldots,i_k\}$ and write $A_{\alpha\beta}$ as M. Applying Lemma 3.11 with $v = -[\epsilon^{i_1}\ \ldots\ \epsilon^{i_k}]^\top$ will result in

$$\det(\mathcal{A}_{\alpha\beta}) = \det\left(M - \begin{bmatrix}\epsilon^{i_1} & \ldots & \epsilon^{i_1}\\ \vdots & \ddots & \vdots\\ \epsilon^{i_k} & \ldots & \epsilon^{i_k}\end{bmatrix}\right) = \det(M) + \sum_{j=1}^{k}\det([M|_j\, v]). \tag{3.8}$$

To calculate $\det([M|_j\, v])$, we will use the Laplace determinant expansion along the jth column, which yields

$$\det([M|_j\, v]) = -\sum_{h=1}^{k}(-1)^{j+h}\epsilon^{i_h}\det(M_{hj})$$

where M_{hj} denotes the matrix obtained from M upon deleting the hth row and jth column. Therefore, we obtain

$$\det(\mathcal{A}_{\alpha\beta}) = \det(M) - \sum_{h=1}^{k}\epsilon^{i_h}\sum_{j=1}^{k}(-1)^{j+h}\det(M_{hj}) = \det(M^\top) + \sum_{h=1}^{k}\epsilon^{i_h}\big(-\det([M^\top|_h\mathbf{1}])\big).$$

For sufficiently small ϵ, we have that $\det(\mathcal{A}_{\alpha\beta})$ is positive (negative) if and only if the vector

$$\Big[\det(M)\quad -\det([M|_1\,\mathbf{1}])\quad \ldots\quad -\det([M|_k\,\mathbf{1}])\Big]$$

is lexico-positive (lexico-negative); hence

$$\zeta(M) = \mathrm{sign}\big(\det(\mathcal{A}_{\alpha\beta})\big).$$

The same calculation can be done for $\det(\mathcal{B}_{\alpha\beta})$. Thus we have established that the index of (x, y) equals

$$(-1)^{t+1}\zeta(A^\top_{\alpha\beta})\zeta(B_{\alpha\beta})$$

as asserted. □

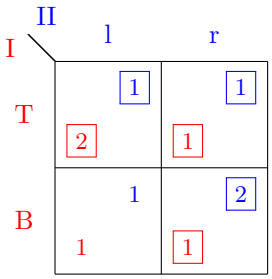
Proposition 3.9 states that there is a one-to-one correspondence between the lexico-feasible and complementary pairs of bases and the equilibria of the lexicographically perturbed game. Given an extreme equilibrium of the original system, enumerating all of those pairs of bases will lead us to all equilibria of the perturbed game that are close to the original equilibrium. The sum of the indices of these equilibria of the perturbed game provides a measure of the contribution of the extreme equilibrium to the index of its component. This motivates the definition of the lex-index.

Definition 3.12 (The lex-index)**.** Let (x, y) be an extreme equilibrium of a bimatrix game (A, B), and let $\mathcal{B}(x, y)$ be the set of all lexico-feasible and complementary pairs of bases that correspond to (x, y). The *lex-index* of (x, y) is defined as (using the same notation as in Theorem 3.10)

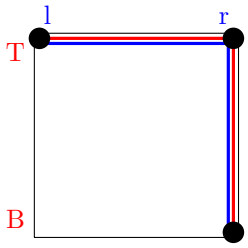
$$\sum_{\mathcal{B}(x,y)}(-1)^{t+1}\zeta(A^\top_{\alpha\beta})\zeta(B_{\alpha\beta}).$$

That is to say, the lex-index of an equilibrium equals the sum of the indices of all equilibria that are near it in a lexicographically perturbed game.

Figure 3.3: Bimatrix game for Example 3.13.



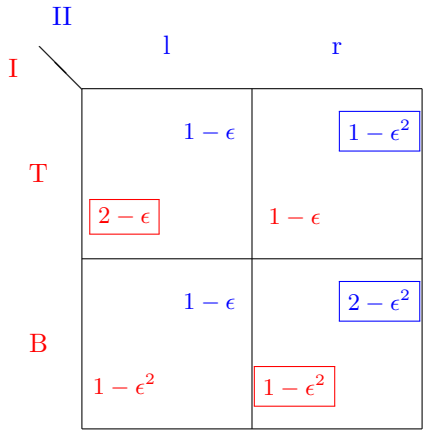
Matrix layout of a degenerate bimatrix game



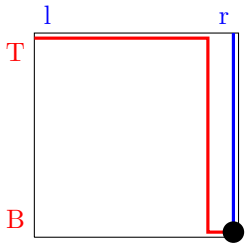
Best response correspondence diagram; black circles represent extreme equilibria.

Example 3.13. Consider the following game, already discussed in Example 2.15.
As explained earlier, this game has three equilibria which form a single equilibrium component, which obviously has index 1 (all indices should add up to +1). Let us analyse the lexicographically perturbed game. Lexicographic perturbation of this game is depicted in Figure 3.4.

Figure 3.4: Lexicographic perturbation.



Matrix layout of a non-degenerate lexicographically perturbed bimatrix game



Best response correspondence diagram; black circles represent extreme equilibria.

Out of the three equilibria of the original game only one 'survived' the perturbation. We should expect that the two equilibria that vanished will have lex-indices equal to 0, while the 'survivor' will have lex-index equal to 1.
If we enumerate all pairs lexico-feasible bases for each equilibrium the only pair of bases that is complementary is $[v, x_2, s_1], [u, y_1, r_2]$ and corresponds to the pure equilibrium (B, r). The corresponding matrices used in (3.7) are 1×1 matrices with a positive entry, so the lex-index of this equilibrium is clearly 1.

Since an equilibrium component can be specified by the extreme equilibria of which it is composed, we have the following result.

Corollary 3.14. *The index of an equilibrium component is the sum of the lex-indices of the extreme equilibria defining that component.*

By now we have accomplished all we need for the description of the algorithm for index computation we shall present in the next chapter. In conclusion, Theorem 3.10 introduces a method to calculate the index of a 'nearby' equilibrium of the lexicographically perturbed game. Using the lex-index approach and Corollary 3.14 we can simply add those indices up and obtain the index of a given equilibrium component.

Chapter 4

Algorithm for index computation

4.1 Introduction

In this chapter we present and explain an algorithm for computing the indices of equilibrium components in degenerate bimatrix games. The algorithm takes an equilibrium component as input and outputs its index. Before detailing the algorithm, let us first consider the bigger picture and describe how one can obtain such input.

Recall that a bimatrix game is specified by two payoff matrices, A and B. Given A and B, we wish to list all extreme equilibria of the game. This problem boils down to enumerating all vertices of the best response polytopes and outputting all pairs of points (i.e. strategies) that satisfy the complementarity condition. Avis et al. (2010) proposed several elegant solutions to the problem. One method is called *lexicographic reverse search* (abbreviated 'lrs'), and we shall use its implementation by Avis et al. in our Program 4.3. In a nutshell, for each polytope the algorithm constructs a 'tree' rooted at 'zero', a known vertex of the polytope that corresponds to the illicit strategy of empty support; this point maximizes the objective function $-\mathbf{1}x^\top$. A lexicographic pivoting rule can determine a unique path from each vertex to 'zero'. These paths are the directed edges of the tree. The reverse search approach traverses these edges in the opposite direction, beginning at the root (i.e. the endpoint of all paths) and going to each possible starting point. For more details see Algorithm 3 in Avis et al. (2010).

Once we have the list of all equilibria, we need to divide it into connected components, i.e. equilibrium components. In order to do so, we need to construct the bipartite graph demonstrated in Example 2.15. The two disjoint sets of vertices of the graph are the two sets of points that correspond to equilibrium strategies of the two players. Vertices are joined by an edge if they form an equilibrium (i.e. satisfy complementarity). The problem is then to find maximal cliques of that graph, that is, to find all maximal complete bipartite sub-graphs (maximal Nash subsets) and return their inclusion-wise maxima. A neat clique enumeration algorithm can be found in Bron and Kerbosch (1973). As described in Avis et al. (2010, Algorithm 2), a modified version of that algorithm fits our bipartite instance. The combined output of the lrs and clique algorithms serves as input to the following algorithm.

4.2 The algorithm

We first present the algorithm of Balthasar (2009) and then explain the three stages.

Input: equilibrium component C of a bimatrix game (A, B).

Output: index of C.

INDEX(C)

$E \leftarrow$ {all extreme equilibria of C};
$i \leftarrow 0$;
for all (x, y) in E **do**
    $i = i +$ *LEX-INDEX*(x, y);
Return i;

LEX-INDEX(x, y)

$B \leftarrow$ *BASES*(x, y);
$i \leftarrow 0$;
for all (α, β) in B **do**
    $k \leftarrow$ the number of x_i that are basic in α;
    $i = i + (-1)^k \zeta(A^{\top}_{\alpha\beta})\zeta(B_{\alpha\beta})$; ▷ Using same notation as in Definition 3.12.
Return i;

BASES(x, y)

$B \leftarrow \emptyset$;
$B_x \leftarrow$ {all lexico-feasible bases that correspond to x};
$B_y \leftarrow$ {all lexico-feasible bases that correspond to y};
for all b_x in B_x **do**
    for all b_y in B_y **do**
        if b_x and b_y are complementary **then**
            $B \leftarrow (b_x, b_y)$;
Return B;

The algorithm consists of three parts:

1. **INDEX** The index of an equilibrium component is calculated as the sum of the lex-indices of the extreme equilibria defining that component (Corollary 3.14).
2. **LEX-INDEX** The lex-index of an extreme equilibrium is obtained by summing the indices of equilibria that are close to it in a lexicographically perturbed game (Definition 3.12); these are the equilibria that correspond to lexico-feasible and complementary pairs of bases. Using the formula in Theorem 3.10, we can calculate these indices without actually perturbing the game.
3. **BASES** There is one-to-one correspondence between the lexico-feasible and complementary pairs of bases and the equilibria of the lexicographically perturbed game (Proposition 3.9).

4.3 Python implementation

This section includes Python code for the core of a program implementing the algorithm presented in the previous section. This implementation does not profess to be efficient by any means. The

programming language Python and the code-writing style were chosen to enhance readability, but in doing so we have compromised on efficiency. We will point out where improvements in running time can be achieved.

The program is based on object-oriented design. We define the following classes:

- EquilibriumComponent
- Equilibrium
- Strategy
- Basis
- PairOfLexicoFeasibleBases

The flow of the program is as follows. As explained in Section 4.1, the input to the program is a combination of the outputs of 'lrs' and 'clique' algorithms (implemented by D. Avis and B. von Stengel respectively). Each equilibrium of the game is specified by a pair of mixed strategies (where a mixed strategy of each player is a probability distribution over the set of pure strategies of that player).

During the initialization phase, we create a *Strategy* object for equilibrium strategy and an *Equilibrium* object for each pair of strategies that defines an equilibrium. Then we process the output of the clique algorithm. For each connected component we create an *EquilibriumComponent* object and define a list of *Equilibria* (from the list we initialized earlier) as a field of that object. This concludes the initialization phase.

Next, we loop over all equilibrium components and invoke the function 'index' of each component object. This triggers a call to the function 'lex-index' of each equilibrium. The lex-index method first fetches all lexico-feasible bases of both strategies (we will get back to how this is done). For each pair of such bases, we create a *PairOfLexicoFeasibleBases* object, check whether the pair satisfies the complementarity condition and calculate its sign (i.e. the index of the corresponding equilibrium of the lexicographically perturbed game given by the formula in Theorem 3.10) using the function 'sign'.

How do we enumerate all the lexico-feasible bases of a strategy? Currently by exhaustive search. If we could find a more efficient method for this task, we would be able to decrease the running time considerably. For each strategy we first generate the set of all possible bases that may correspond to the strategy by creating a list of *Basis* objects. For each of these bases, we check if it is feasible and lexico-positive.

```
from numpy.linalg import inv as inverse, slogdet as determinant_sign, matrix_rank
from itertools import combinations as all_subsets
from fractions import Fraction
import json
from numpy import zeros

class EquilibriumComponent:
    def __init__(self, extreme_equilibria, nash_subsets):
        self.extreme_equilibria = extreme_equilibria
        self.nash_subsets = nash_subsets

    def index(self):
        index = 0
        for equilibrium in self.extreme_equilibria:
            index += equilibrium.lex_index
        return index

class Equilibrium:
    # x and y are of type Strategy
    def __init__(self,x, y):
        self.x = x
        self.y = y
        self.lex_index = self.get_lex_index()

    def get_lex_index(self):
        lex_index = 0
        x_bases = self.x.lexico_feasible_bases()
        y_bases = self.y.lexico_feasible_bases()
        for alpha in x_bases:
            for beta in y_bases:
                pair = PairOfLexicoFeasibleBases(alpha, beta)
                if pair.fulfils_complementarity():
                    lex_index += pair.sign()
        return lex_index

class Strategy:
    def __init__(self, player, distribution, payoff, number):
        self.player = player
        self.distribution = distribution
        self.payoff = payoff
        self.number = number

    def support(self):
        result = []
        for index, strategy in enumerate(self.distribution):
            if strategy > 0:
                result.append(index)
        return result

    def lexico_feasible_bases(self):
        # u/v are always basic; offset +1 to support indices
        basic_variables = [0] + [i+1 for i in self.support()]
        # 'basis_size' will be n+1 for player 1 and m+1 for player 2
        opponent_number_of_pure_strategies = n if self.player == 1 else m
        basis_size = opponent_number_of_pure_strategies + 1
        # 'how_many_to_add' gets the number of variables we need to add to form a basis
```

```
        how_many_to_add = basis_size - len(basic_variables)
        # 'candidates_to_add' gets all possible variables we can add to a basis
        candidates_to_add = \
            [item for item in range(1, m+n+1) if item not in basic_variables]
        # 'subsets' gets all subsets of 'candidates_to_add' of size 'how_many_to_add'
        subsets = all_subsets(candidates_to_add, how_many_to_add)
        all_bases = []
        for indices_to_add in subsets:
            indices = sorted(basic_variables + list(indices_to_add))
            all_bases.append(Basis(indices, self))

        lexico_feasible_bases = \
            [basis for basis in all_bases if basis.is_lexico_feasible()]
        return lexico_feasible_bases

class Basis:
    def __init__(self, indices, strategy):
        self.indices = indices
        self.strategy = strategy
        self.player = self.strategy.player
        self.number_of_pure_strategies = m if self.player == 1 else n
        self.matrix = matrix_1 if self.player == 1 else matrix_2
        self.basic_matrix_inverse = self.get_basic_matrix_inverse()

    def get_basic_matrix_inverse(self):
        basic_matrix = self.matrix[:, self.indices]
        if Basis.is_singular(basic_matrix):
            return None
        else:
            return inverse(basic_matrix)

    def is_lexico_feasible(self):
        if self.not_a_basis():
            return False
        elif self.infeasible():
            return False
        elif self.solution_does_not_correspond_to_strategy():
            return False
        else:
            return self.is_lexico_positive()

    def not_a_basis(self):
        return self.basic_matrix_inverse == None

    def infeasible(self):
        return min(self.basic_variables_vector()) < 0

    def solution_does_not_correspond_to_strategy(self):
        strategy_variables = self.basic_solution()[1:self.number_of_pure_strategies+1]
        for i, item in enumerate(strategy_variables):
            # checking whether strategy distribution is equal to basic solution
            if item != round(float(self.strategy.distribution[i]), 5):
                return True

        return False
```

```
    def is_lexico_positive(self):
        flag = True
        dimension = self.basic_matrix_inverse.shape[0]
        for row in range(dimension):
            if flag == False:
                break
            else:
                for column in range(dimension):
                    current = round(self.basic_matrix_inverse[row][column], 5)
                    if current == 0: # move to next coordinate if 0
                        continue
                    elif current < 0: # lexico-negative if < 0
                        flag = False
                        break
                    else: # check next row
                        break
        return flag

    def basic_variables_vector(self):
        # basic variables vector will be the first column of the basic inverse matrix
        return [round(row[0],5) for row in self.basic_matrix_inverse]

    def basic_solution(self):
        basic_variables_vector = self.basic_variables_vector()
        result = []
        j = 0
        # m+n+1 is the number of components in the variable vector
        for i in range(m+n+1):
            if i in self.indices:
                result.append(basic_variables_vector[j])
                j = j+1
            else:
                result.append(0)

        return result

    def basic_strategy_variables(self):
        result = []
        for i in range(1, self.number_of_pure_strategies + 1):
            if i in self.indices: result.append(i-1)
        return result

    @staticmethod
    def is_singular(matrix):
        return not Basis.is_square(matrix) or not Basis.is_full_rank(matrix)

    @staticmethod
    def is_square(matrix):
        return matrix.shape[0] == matrix.shape[1]

    @staticmethod
    def is_full_rank(matrix):
        return matrix_rank(matrix) == matrix.shape[0]

class PairOfLexicoFeasibleBases:
    def __init__(self, alpha, beta):
```

```
        self.alpha = alpha
        self.beta = beta

    def fulfils_complementarity(self):
        flag = True
        for i in range(1, m+1):
            if i in self.alpha.indices and i+n in self.beta.indices:
                flag = False
                break
        if flag:
            for i in range(1, n+1):
                if i in self.beta.indices and i+m in self.alpha.indices:
                    flag = False
                    break
        return flag

    def square_submatrix(self, matrix):
        alpha_indices = self.alpha.basic_strategy_variables()
        beta_indices = self.beta.basic_strategy_variables()
        return matrix[alpha_indices,:][:,beta_indices]

    def sign(self):
        t = len(self.alpha.basic_strategy_variables())
        sign_of_A = sign_of_matrix(self.square_submatrix(A).transpose())
        sign_of_B = sign_of_matrix(self.square_submatrix(B))
        return (-1)**(t+1) * sign_of_A * sign_of_B

def sign_of_matrix(matrix):
    dimension = matrix.shape[0]
    sign = determinant_sign(matrix)[0] # get the sign of the determinant of 'matrix'
    i = 0
    while sign == 0 and i < dimension:
        sign = (-1) * determinant_sign(replace_column_by_1(matrix, i))[0]
        i = i+1

    return sign

def replace_column_by_1(matrix, index):
    clone = copy(matrix)
    clone[:, index] = 1 # replace column 'index' by the vector 1.
    return clone
```

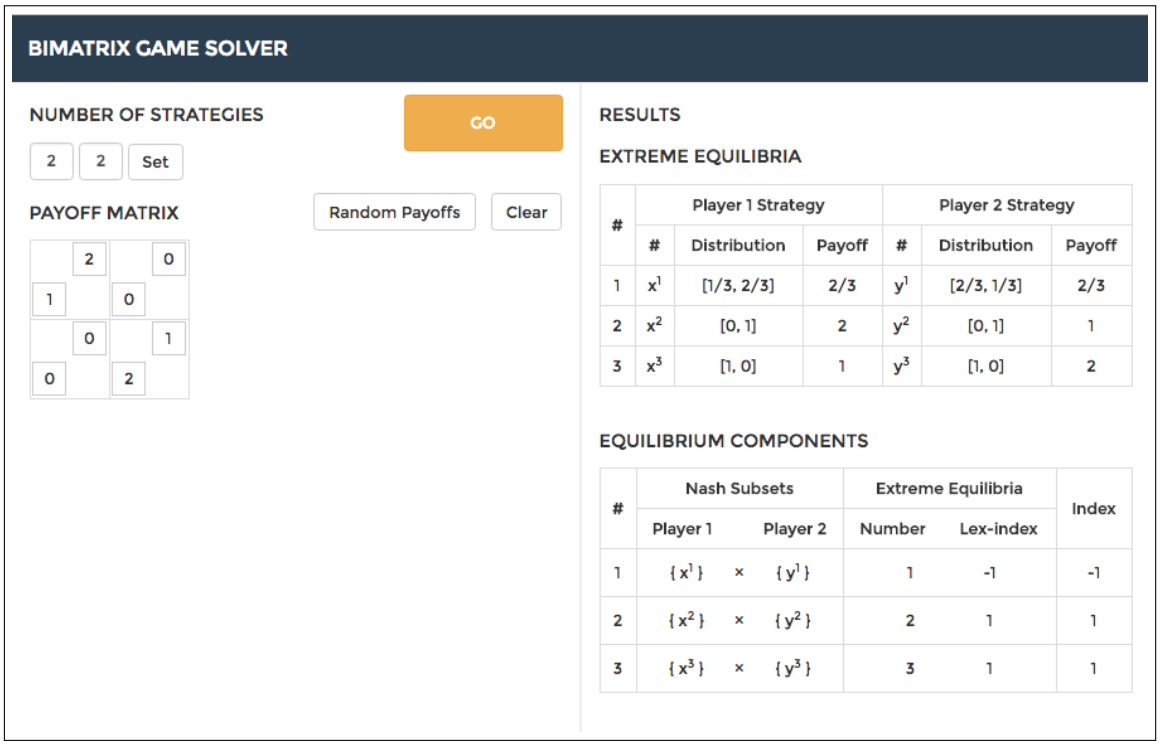
4.4 Web interface

In order to make our program easily accessible we have developed a web-interface for it. Hosting an application on a web server spares the hassle of installation and allows the option to deploy updates constantly. The application is available at the following url.

`https://bimatrix.herokuapp.com`

This an is initial release that is planned to be improved and extended in the near future. We present a screenshot followed by usage instructions.

Figure 4.1: Screenshot of output for the 'Battle of Sexes' game presented in Example 3.2



To use the program one has to choose the number of strategies of the game and thereupon fill in its payoff matrices. This can be done using the forms on the left-hand side of the page. Pressing the 'GO' button will submit the request to the server which will solve the submitted game.
The results will appear in the right-hand side and can be read as follows. The upper table consists of a list of all extreme equilibria of the game. For each equilibrium we present the strategies of the two player and assign it a number. For each equilibrium strategy we present its probability distribution and payoff to the respective player and assign it a number.
The lower table contains information about the equilibrium components. For each component we present a list of its 'maximal Nash subsets', (the convex sets whose union yields the component), a list of all extreme equilibria of the component side by side with their lex-indices, and, of course, the index of the component. Nash subsets are displayed as product of two sets containing strategies (represented by their numbers) of the two player. The numbering of extreme equilibria of a component corresponds to the numbering in the list of all equilibria.

4.5 Equilibrium components with arbitrary index

Recall that an equilibrium component's index can take on any integer value. Govindan et al. (2003) showed that for each integer q there exists a bimatrix game that has an equilibrium component with index q and presented a method to construct such a game (Govindan et al., 2003, Proposition 3.1). We will follow this game construction method and present the program output of it (for more information about the construction see Govindan et al. (2003, Chapter 3)). Consider the following 3×3 game.

I \ II	l	c	r
T	13, 13	12, 7	**14**, 1
M	7, 12	**8**, **8**	1, **2**
B	1, **14**	**2**, 1	1, 1

Figure 4.2: Matrix layout; the pure best response payoffs are enclosed in squares.

Submitting this game results the following output. Let us analyse its equilibria structure.

BIMATRIX GAME SOLVER

NUMBER OF STRATEGIES: 3 3 Set — GO

PAYOFF MATRIX — Random Payoffs — Clear

13, 13	12, 7	14, 1
7, 12	8, 8	1, 2
1, 14	2, 1	1, 1

RESULTS

EXTREME EQUILIBRIA

#	Player 1 Strategy			Player 2 Strategy		
	#	Distribution	Payoff	#	Distribution	Payoff
1	x^1	[1/2, 1/12, 5/12]	15/2	y^1	[1/2, 1/12, 5/12]	15/2
2	x^2	[1/2, 1/2, 0]	10	y^2	[1/2, 1/2, 0]	10
3	x^3	[0, 1, 0]	8	y^3	[0, 1, 0]	8

EQUILIBRIUM COMPONENTS

#	Nash Subsets		Extreme Equilibria		Index
	Player 1	Player 2	Number	Lex-index	
1	$\{x^1\}$	$\{y^1\}$	1	1	1
2	$\{x^2\}$	$\{y^2\}$	2	-1	-1
3	$\{x^3\}$	$\{y^3\}$	3	1	1

As we can see in the above screenshot, this game is non-degenerate and has three equilibria. Next, we introduce a fourth strategy to each player called an 'outside option'. This strategy gives the same constant payoff to both players and can be thought of an earlier move that a player can

choose to take. If no player chooses the outside option then the original game is played. We will add an outside option of payoff 9 to both players; this will cause any equilibrium of the original game with payoff of less than 9 to disappear in the resulting game. In our case, only equilibrium #2 will remain in the game (and will keep its index). Notice that this equilibrium has index -1. Since all indices should add up to +1 we except the sum of remaining indices (of other equilibrium components) to be 2. The resulting game will be the following.

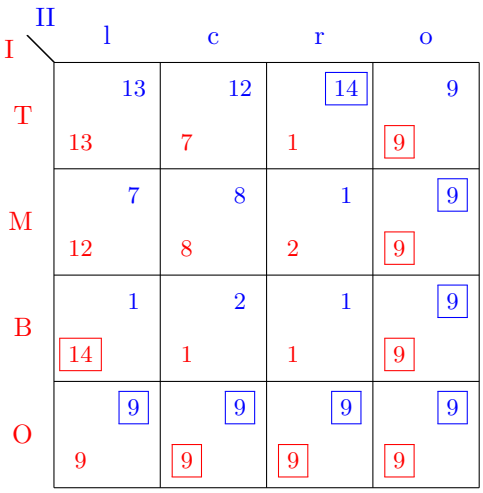


Figure 4.3: Matrix layout; addition of an outside option to both players.

This game is clearly degenerate. Notice the pure equilibrium where both players choose the outside option; this means that any equilibrium where one of the player plays the outside option (purely) will be in the same connected component. In our case, this is the only component besides the isolated equilibrium we found earlier. The program output is the following.

EXTREME EQUILIBRIA

#	Player 1 Strategy			Player 2 Strategy		
	#	Distribution	Payoff	#	Distribution	Payoff
1	x^1	[0, 0, 0, 1]	9	y^1	[1/4, 3/4, 0, 0]	9
2	x^1	[0, 0, 0, 1]	9	y^2	[0, 1, 0, 0]	9
3	x^1	[0, 0, 0, 1]	9	y^3	[1/2, 1/3, 1/6, 0]	9
4	x^1	[0, 0, 0, 1]	9	y^4	[8/13, 4/39, 11/39, 0]	9
5	x^1	[0, 0, 0, 1]	9	y^5	[8/13, 0, 5/13, 0]	9
6	x^1	[0, 0, 0, 1]	9	y^6	[0, 0, 1, 0]	9
7	x^1	[0, 0, 0, 1]	9	y^7	[0, 0, 0, 1]	9
8	x^2	[8/13, 4/39, 11/39, 0]	9	y^7	[0, 0, 0, 1]	9
9	x^3	[8/13, 0, 5/13, 0]	9	y^7	[0, 0, 0, 1]	9
10	x^4	[0, 0, 1, 0]	9	y^7	[0, 0, 0, 1]	9
11	x^5	[1/2, 1/3, 1/6, 0]	9	y^7	[0, 0, 0, 1]	9
12	x^6	[1/4, 3/4, 0, 0]	9	y^7	[0, 0, 0, 1]	9
13	x^7	[0, 1, 0, 0]	9	y^7	[0, 0, 0, 1]	9
14	x^8	[1/2, 1/2, 0, 0]	10	y^8	[1/2, 1/2, 0, 0]	10

EQUILIBRIUM COMPONENTS

#	Nash Subsets			Extreme Equilibria		Index
	Player 1		Player 2	Number	Lex-index	
1	$\{x^1, x^2, x^3, x^4, x^5, x^6, x^7\}$	×	$\{y^7\}$	1	0	2
				2	0	
				3	1	
				4	0	
				5	0	
				6	0	
				7	0	
	$\{x^1\}$	×	$\{y^1, y^2, y^3, y^4, y^5, y^6, y^7\}$	8	0	
				9	0	
				10	0	
				11	1	
				12	0	
				13	0	
2	$\{x^8\}$	×	$\{y^8\}$	14	-1	-1

We can tell that the original equilibrium 'survived' (numbered 14 in the above output) and its index remained -1. The other equilibrium component consists of 2 maximal Nash subsets that share the pure strategy equilibrium where both players play the outside option. It has index 2 as expected. Using this method one can construct bimatrix games that have components with arbitrary index. We were able verify this method of construction for the 7×7 and 10×10 games that produce components with indices 3 and 4 respectively. Since the output is very long we will not present it here.

Bibliography

Avis D., Rosenberg G., Savani R. and von Stengel B. (2010), Enumeration of Nash equilibria for two-player games. *Economic Theory* 42, pp. 9–37. DOI: 10.1007/s00199-009-0449-x.

Balthasar A. (2009), Geometry and equilibria in bimatrix games. *PhD Thesis*, The London School of Economics and Political Science.

Bron C. and Kerbosch J. (1973), Algorithm 457: Finding all cliques of an undirected graph. *Comm. ACM* 16, pp. 575–577.

Dantzig G. B. (1963), *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, pp. 153–155.

Govindan S., von Schemde A. and von Stengel B. (2003), Symmetry and p-stability. *International Journal of Game Theory*, 32, pp. 359–369. DOI: 10.1007/s001820400167

Jones C. N., Kerrigan E. C. and Maciejowski J. M. (2007), Lexicographic perturbation for multi-parametric linear programming with applications to control. *Automatica* 43(10), pp. 1808–1816.

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

Nash J. (1951), Non-cooperative games. *Annals of Mathematics* 54, pp. 286–295.

Shapley L. S. (1974), A note on the Lemke-Howson algorithm. In: *Pivoting and Extension*, Mathematical Programming Study 1, North-Holland, Amsterdam, pp. 175–189.

von Schemde A. and von Stengel B. (2008), Strategic characterization of the index of an equilibrium. In: *Proceedings of the 1st International Symposium on Algorithmic Game Theory (SAGT 2008)*, eds B. Monien and U.-P. Schroeder, Lecture Notes in Computer Science vol. 4997, Springer, Berlin, pp. 242–254.

von Stengel B. (1996), Computing equilibria for two-person games. *Technical Report* 253, Department of Computer Science, ETH Zürich.