RELAXATION ALGORITHMS IN FINDING NASH EQUILIBRIA

STEFFAN BERRIDGE AND JACEK B. KRAWCZYK

ABSTRACT. Relaxation algorithms provide a powerful method of finding noncooperative equilibria in general synchronous games. Through use of the Nikaido-Isoda function, the Nash solution to a broad category of constrained, multiplayer, non-zerosum games can easily be found. We provide solutions to some simple games using this procedure and extend ourselves to more difficult games involving coupled constraints and multiple discrete time periods using a program developed in Matlab.

Working Paper

Version 2.3

JEL Keywords: C63 (Computational Techniques), C72 (Noncooperative Games), C87 (Economic Software), E62 (Taxation), Q25 (Water; Air).

AMS Keywords: 90-08 (Computational Methods), 90A14 (Equilibrium: general theory), 90A30 (Environmental economics), 90A56 (Special types of equilibria), 90A58 (Models of real-world systems), 90A70 (Macroeconomic policy-making, taxation), 90D05 (2-person games), 90D06 (*n*-person games), 90D10 (Noncooperative games), 90D50 (Discrete-time games), 90D80 (Applications of game theory).

Authors' Keywords: Computational economics; Nash normalised equilibrium, coupled constraints, Nikaido-Isoda function, open-loop Nash equilibrium.

Research supported by VUW GSBGM and IGC.

Correspondence should be addressed to: Jacek B. Krawczyk.

About the authors: STEFFAN BERRIDGE. Institute of Statistics and Operations Research (*Masters' Student*), Victoria University of Wellington. Email: Steffan Berridge@vuw.ac.nz ;
http://www.vuw.ac.nz/ berridge

JACEK B. KRAWCZYK. Commerce & Administration Faculty, Victoria University of Wellington, PO Box 600, Wellington, New Zealand; fax +64-4-4712200. Email: Jacek.Krawczyk@vuw.ac.nz ;
http://www.vuw.ac.nz/ jacek

CONTENTS

1. Introduction 3
2. Definitions and concepts 3
2.1. Conventions 3
2.2. Nash equilibrium and the Nikaido-Isoda function 4
3. The relaxation algorithm 7
3.1. Statement of the algorithm 7
3.2. Conditions for existence of a Nash equilibrium and convergence of relaxation algorithm 8
3.3. Step size optimisation 8
4. Some simple examples with static games 9
4.1. Simple two player games 9
4.2. A static game with coupled constraints 12
5. Finding equilibria in dynamic games 19
5.1. Application to dynamic games 19
5.2. Formulation of the game 19
5.3. Solution to the game 20
6. Conclusions 26
Appendix A. Notes on customarily designed Matlab software for solutions of games 27
A.1. Program design 27
A.2. Description of the procedures 27
References 29

Relaxation Algorithms in finding Nash Equilibria

1. Introduction

Game theory problems are renown for being difficult to solve, especially many player, non-zero sum and dynamic games. Computation of Nash equilibria using the Nikaido-Isoda function [4] and the relaxation algorithm [2], [8] seems to be an attractive possibility in that the most advanced technique required is minimisation of a multivariate function; a well studied topic. The aim of this paper is to use the relaxation algorithm to find Nash equilibria in games of varying complexity.

The motivating paper [8] sets the foundation for the usage of the relaxation algorithm to find Nash normalised equilibria, and promises its success for a certain class of games. The contribution of the present paper is in applyinging this idea to some specific game theory problems and reporting on equilibrium computation experiments. The experiments below were conducted using customarily developed software in the Matlab programming environment.

In the latter stages of the paper, we are able to solve coupled constraints games, which force players to obey constraints based on their collective actions. Obeying such constraints relies on cooperation, however taxes can be enforced to ensure that players meet these constraints (see [3].) We solve a coupled constraints game in a static and then a dynamic setting using the open loop information pattern. As analytical solutions to such games hardly ever exist, the strength of this paper is in studying a new *computational-economics* method suitable for finding equilibria in intertemporal ("open-loop") conflict situations.

The organisation of the paper is as follows. Sections 2 and 3 have a tutorial character and provide an introduction to some elementary concepts. In Section 4, we consider some simple examples which provide an insight into how the algorithm works. Section 5 applies the relaxation algorithm to a dynamic game. Concluding remarks and an Appendix, which describes the software we developed, are at the end of the paper.

All definitions, theorems and such are numbered communally and consecutively in each section.

2. Definitions and concepts

2.1. **Conventions.**

Convention 2.1. *In this paper we will be considering two different levels of vector; each player in a game will have an* action*, and all players together will have a* collective action*. To avoid notational confusion between the two and offer the reader some comfort, we will adopt the convention that a player's action will be in normal type and have a subscript (*e.g. $x_i \in \boldsymbol{R}^{m_i}$*), and the collective action will be in boldface (*e.g. $\mathbf{x} \in \boldsymbol{R}^{m_1} \times \cdots \times \boldsymbol{R}^{m_n}$*). In this notation, $\mathbf{x} = (x_1, \ldots, x_n)$.*

Convention 2.2. *Political correctness is a theme of this decade, and singular pronouns are often required to denote a non gender specific person. We adopt the convention that the words "he" and "she" have the common meaning "he or she" where appropriate.*

2.2. **Nash equilibrium and the Nikaido-Isoda function.** This paper concerns game theory, we therefore present some basic definitions in this area.

Definition 2.3. *A* game in normal form *is a three-tuple* $\{N, (X_i)_{i\in N}, (\phi_i)_{i\in N}\}$ *where* N *is the set of players,* $N = \{1, 2, \ldots, n\}$*,* X_i *is the action space of player* i *and* ϕ_i *is the payoff function of player* i*,* $\phi_i : X_1 \times X_2 \times \ldots \times X_n \to \boldsymbol{R}$*.*

Notation 2.4. *Let* $\mathbf{x} = (x_1, \ldots, x_n)$ *and* $\mathbf{y} = (y_1, \ldots, y_n)$ *be elements of the collective action space* $X_1 \times \ldots \times X_n$*. Denote the element* $(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$ *by* $(y_i|\mathbf{x})$*.*

Definition 2.5. *Let* $X \subset \boldsymbol{R}^{m_1} \times \ldots \times \boldsymbol{R}^{m_n} = \boldsymbol{R}^m$ *be the collective action space, and the function* $\phi_i : X \to \boldsymbol{R}$ *be the payoff function for player* i*,* $i \in N$*. Then the point* $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ *is called the* Nash equilibrium point *if, for each* i*,*

$$\phi_i(\mathbf{x}^*) = \max_{x_i \in \boldsymbol{R}^{m_i}} \phi_i(x_i|\mathbf{x}^*). \tag{1}$$

We will use the notation

$$\mathbf{x}^* = \arg\operatorname{equil}_{\mathbf{x}\in\boldsymbol{R}^m} \left\{ (X_i)_{i\in N}, (\phi_i)_{i\in N} \right\}$$

to denote the equilibrium point of the game where player i *has action space and payoff functions* X_i *and* ϕ_i *respectively.*

We now introduce the Nikaido-Isoda function [8]. This function is not so elementary in the realms of game theory, and bears some thinking about.

Definition 2.6. *Let* ϕ_i *be the payoff function for player* i*. Then the* Nikaido-Isoda function $\Psi : (X_1 \times \cdots \times X_n) \times (X_1 \times \cdots \times X_n) \to \boldsymbol{R}$ *is defined as*

$$\Psi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} [\phi_i(y_i|\mathbf{x}) - \phi_i(\mathbf{x})]. \tag{2}$$

Result 2.7. [8]

$$\Psi(\mathbf{x}, \mathbf{x}) \equiv 0 \qquad\qquad \mathbf{x} \in X. \tag{3}$$

Let us take a brief look at the significance of this function. Each summand can be thought of as the improvement in payoff a player will receive by changing his action from x_i to y_i while all other players continue to play according to $\mathbf{x}$. The function thus represents the sum of these improvements in payoff. Note that the *maximum* value this function can take, for a given $\mathbf{x}$, is always nonnegative, owing to Result 2.7 above. Also the function is everywhere nonpositive when either $\mathbf{x}$ or $\mathbf{y}$ is a Nash equilibrium point, since in an equilibrium situation no player can make a unilateral improvement to their payoff, and so each summand in this case can be at most zero.

From here, we reach the conclusion that when the Nikaido-Isoda function cannot be made (significantly) positive for a given $\mathbf{y}$, we have (approximately) reached the Nash equilibrium point. We use this observation in constructing a termination condition for our algorithm; that is, we choose an ε such that, when $\max_{\mathbf{y}\in\mathbf{R}^m} \Psi(\mathbf{x}^s, \mathbf{y}) < \varepsilon$, we have achieved the equilibrium to a sufficient degree of precision.

Definition 2.8. *An element* $\mathbf{x}^* \in X$ *is referred to as a* Nash normalised equilibrium point *if*

$$\max_{\mathbf{y} \in X} \Psi(\mathbf{x}^*, \mathbf{y}) = 0. \tag{4}$$

Notice that Rosen [6] also introduces the notion of a normalised equilibrium when examining equilibrium point uniqueness in concave n-person games. However, he defines it in a different context which is one of a game with coupled constraints. His definition will be presented when we deal with coupled constraints games in Section 4.2.

The two following lemmas are presented in [1]:

Lemma 2.9. *A Nash normalised equilibrium point is also a Nash equilibrium point.*

Lemma 2.10. *A Nash equilibrium point is a Nash normalised equilibrium point if the collective action space X satisfies*

$$X = X_1 \times \ldots \times X_n \qquad\qquad X_i \subset \boldsymbol{R}^{m_i}. \tag{5}$$

An algorithm which uses the Nikaido-Isoda function to find the Nash normalised equilibrium will be developed in the next section. Here we note that at each point of the algorithm we wish to move towards a point which is an "improvement" on the one that we are at. To this end, let us put forward the following definition.

Definition 2.11. *The* optimum response function *at point* $\mathbf{x}$ *is*

$$Z(\mathbf{x}) = \arg\max_{\mathbf{y} \in X} \Psi(\mathbf{x}, \mathbf{y}). \tag{6}$$

In brief terms, this function returns the set of players' actions whereby they all attempt to unilaterally maximise their payoffs.

It is interesting to note that we can consider the algorithm as either performing a static optimisation or as calculating successive actions in a convergence to equilibrium in a real time process. We have been considering the case where all payoffs are known to us, in which case we can directly find the Nash equilibrium using the relaxation algorithm. However, if we only had access to one player's payoff function and all players' past actions, then at each stage in the real time process we choose the optimum response for that player, assuming that the other players will play as they had in the previous period. In this way, convergence to the Nash normalised equilibrium will occur as $t \to \infty$.

We now introduce some more technical definitions to be used in the main theorem.

Definition 2.12. *A function of one argument* $f(\mathbf{x})$ *is* weakly convex *if,* $\forall \mathbf{x}, \mathbf{y} \in X$

$$\begin{gathered} \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) \geq f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) + \alpha(1-\alpha) r(\mathbf{x}, \mathbf{y}) \\ 0 \leq \alpha \leq 1, \text{ and } \frac{r(\mathbf{x}, \mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|} \to 0 \text{ as } \|\mathbf{x} - \mathbf{y}\| \to 0 \ \ \forall \mathbf{x} \in X \end{gathered} \tag{7}$$

Definition 2.13. *A function of one argument* $f(\mathbf{x})$ *is* weakly concave *if,* $\forall \mathbf{x}, \mathbf{y} \in X$

$$\alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) \leq f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) + \alpha(1-\alpha)\mu(\mathbf{x}, \mathbf{y}) \tag{8}$$
$$0 \leq \alpha \leq 1, \text{ and } \frac{\mu(\mathbf{x}, \mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|} \to 0 \text{ as } \|\mathbf{x}-\mathbf{y}\| \to 0 \ \forall \mathbf{x} \in X$$

Definition 2.14. *A function of two vector arguments,* $f(\mathbf{x}, \mathbf{y})$ *is referred to as* weakly convex-concave [8] *if it satisfies weak convexity with respect to its first argument and weak concavity with respect to its second.*

That is, for fixed $\mathbf{z} \in X$*,*

$$\alpha f(\mathbf{x}, \mathbf{z}) + (1-\alpha) f(\mathbf{y}, \mathbf{z}) \geq f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}, \mathbf{z}) + \alpha(1-\alpha) r(\mathbf{x}, \mathbf{y}; \mathbf{z}) \tag{9}$$
$$\mathbf{x}, \mathbf{y} \in X, \ 0 \leq \alpha \leq 1, \text{ and } \frac{r(\mathbf{x}, \mathbf{y}; \mathbf{z})}{\|\mathbf{x}-\mathbf{y}\|} \to 0 \text{ as } \|\mathbf{x}-\mathbf{y}\| \to 0 \ \forall \mathbf{x} \in X$$

and

$$\alpha f(\mathbf{z}, \mathbf{x}) + (1-\alpha) f(\mathbf{z}, \mathbf{y}) \leq f(\mathbf{z}, \alpha \mathbf{x} + (1-\alpha)\mathbf{y}) + \alpha(1-\alpha) \mu(\mathbf{x}, \mathbf{y}; \mathbf{z}) \tag{10}$$
$$\mathbf{x}, \mathbf{y} \in X, \ 0 \leq \alpha \leq 1, \text{ and } \frac{\mu(\mathbf{x}, \mathbf{y}; \mathbf{z})}{\|\mathbf{x}-\mathbf{y}\|} \to 0 \text{ as } \|\mathbf{x}-\mathbf{y}\| \to 0 \ \forall \mathbf{x} \in X$$

$r(\mathbf{x}, \mathbf{y}; \mathbf{z})$ *and* $\mu(\mathbf{x}, \mathbf{y}; \mathbf{z})$ *are referred to as the residual terms.*

The notions of weak convexity and concavity are weakenings of strict convexity and concavity. The residual terms, to be chosen at will, ensure that there are many concave functions which are weakly convex and many convex functions which are weakly concave.

In fact, the family of weakly convex-concave functions includes the family of smooth functions[1] [8], and so this condition is a minimal restriction for practical purposes.

We now present an elementary example to illustrate the above ideas.

Example: *A convex function which is weakly concave*

Consider the function $f : \mathbf{R} \to \mathbf{R}$ defined by

$$f(x) = x^2.$$

This function is convex in all senses. To show that it is weakly concave, we must find an $r(x, y)$ such that, for all $x, y \in \mathbf{R}$ and $\alpha \in [0, 1]$,

$$\alpha f(x) + (1-\alpha) f(y) \leq f(\alpha x + (1-\alpha)y) + \alpha(1-\alpha) r(x, y)$$

That is if and only if

$$\alpha x^2 + (1-\alpha)y^2 \leq (\alpha x + (1-\alpha)y)^2 + \alpha(1-\alpha) r(x, y)$$
$$\Leftrightarrow \quad \alpha x^2 + (1-\alpha)y^2 \leq \alpha^2 x^2 + (1-\alpha)^2 y^2 + 2\alpha(1-\alpha)xy + \alpha(1-\alpha) r(x, y)$$
$$\Leftrightarrow \quad \alpha(1-\alpha)x^2 + \alpha(1-\alpha)y^2 - 2\alpha(1-\alpha)xy \leq \alpha(1-\alpha) r(x, y)$$
$$\Leftrightarrow \quad (x-y)^2 \leq r(x, y)$$

[1] Recall that a function is smooth iff its derivatives of all orders are continuous.

So it is sufficient to select $r(x,y) = (x-y)^2$. Also

$$\frac{r(x,y)}{\|x-y\|} = \frac{(x-y)^2}{|x-y|} = |x-y| \to \quad 0 \qquad\qquad \text{as } |x-y| \to 0.$$

So we conclude that $f(x) = x^2$ is weakly concave. (However, as this is a one variable function, it cannot be weakly convex-concave.) $\diamond$

The function $r(\mathbf{x},\mathbf{y};\mathbf{z})$ was introduced with the concept of weak convex-concavity. In the case of $\Psi(\mathbf{x},\mathbf{y})$ being a twice continuously differentiable function with respect to both arguments on $X \times X$, the residual terms satisfy

$$r(\mathbf{x},\mathbf{y};\mathbf{y}) = \frac{1}{2}\langle A(\mathbf{x},\mathbf{x})(\mathbf{x}-\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle + o_1(\|\mathbf{x}-\mathbf{y}\|^2) \tag{11}$$

and

$$\mu(\mathbf{y},\mathbf{x};\mathbf{x}) = \frac{1}{2}\langle B(\mathbf{x},\mathbf{x})(\mathbf{x}-\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle + o_2(\|\mathbf{x}-\mathbf{y}\|^2) \tag{12}$$

where $A(\mathbf{x},\mathbf{x}) = \Psi_{\mathbf{xx}}(\mathbf{x},\mathbf{y})|_{\mathbf{y}=\mathbf{x}}$ is the Hessian of the Nikaido Isoda function with respect to the first argument and $B(\mathbf{x},\mathbf{x}) = \Psi_{\mathbf{yy}}(\mathbf{x},\mathbf{y})|_{\mathbf{y}=\mathbf{x}}$ is the Hessian of the Nikaido-Isoda function with respect to the second argument, both evaluated at $\mathbf{y} = \mathbf{x}$.

To prove the inequality of Condition (5) of the following Theorem 3.1 under the assumption that $\Psi(\mathbf{x},\mathbf{y})$ is twice continuously differentiable, it suffices to show that $Q(\mathbf{x},\mathbf{x}) = A(\mathbf{x},\mathbf{x}) - B(\mathbf{x},\mathbf{x})$ is strictly positive.

Definition 2.15. *The set X is convex if, for every two points* $\mathbf{x}$ *and* $\mathbf{y}$ *in X, the point* $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ *is also in X for* $0 < \alpha < 1$.

Definition 2.16. *The set X is compact if every Cauchy sequence has a limit in X. In a finite dimensional space this is equivalent to X being closed and bounded.*

3. The relaxation algorithm

3.1. **Statement of the algorithm.** Suppose we wish to find the Nash equilibrium for some game and we have some initial estimate of it, say $\mathbf{x_0}$. The relaxation algorithm we use is

$$\mathbf{x}^{s+1} = (1-\alpha_s)\mathbf{x}^s + \alpha_s Z(\mathbf{x}^s) \qquad 0 < \alpha_s \leq 1 \tag{13}$$

$$s = 0, 1, 2, \ldots$$

The iterate at step $s+1$ is constructed by a weighted average of the improvement point $Z(\mathbf{x}^s)$ and the current point $\mathbf{x}^s$. This averaging ensures convergence of the algorithm under certain conditions, as stated in the following Theorem 3.1.

By taking sufficiently many iterations of the algorithm, it is our aim to determine the Nash equilibrium $\mathbf{x}^*$ with arbitrary precision. Indeed Theorem 3.1 in the following subsection provides conditions for the convergence of this relaxation algorithm.

3.2. **Conditions for existence of a Nash equilibrium and convergence of relaxation algorithm.** The following theorem states the conditions of convergence for the relaxation algorithm. The conditions may look rather restrictive, but in fact a large class of games satisfy them.

Theorem 3.1. [8] *There exists a unique Nash equilibrium point to which the algorithm converges if :*

1. *X is a convex compact subset of $\boldsymbol{R}^m$,*
2. *the Nikaido-Isoda function $\Psi : X \times X \to \boldsymbol{R}$ is a weakly convex-concave function and $\Psi(\mathbf{x},\mathbf{x}) = 0 \ \mathbf{x} \in X$,*
3. *the optimum response function $Z(\mathbf{x})$ is single valued and continuous on X,*
4. *the residual term $r(\mathbf{x},\mathbf{y};\mathbf{z})$ is uniformly continuous on X wrt $\mathbf{z}$,*
5. *the residual terms satisfy*

$$r(\mathbf{x},\mathbf{y};\mathbf{y}) - \mu(\mathbf{y},\mathbf{x};\mathbf{x}) \geq \beta(\|\mathbf{x}-\mathbf{y}\|) \quad \mathbf{x},\mathbf{y} \in X \tag{14}$$

   where β is a strictly increasing function,

6. *the relaxation parameters α_s satisfy*
   (a) $\alpha_s > 0$,
   (b) $\sum_{s=0}^{\infty} \alpha_s = \infty$,
   (c) $\alpha_s \to 0$ *as* $s \to \infty$.

3.3. **Step size optimisation.** In order for the algorithm to converge, we may choose any sequence (α_s) satisfying the final condition of Theorem 3.1. However, it is of computational importance to attempt to optimise the convergence rate.

Suitable step-sizes may be obtained by trial and error, and we have found that using a constant step of $\alpha_s \equiv 0,5$ leads to a quick convergence in most of our experiments. In this case, we can think of the α_s as being constant until our convergence conditions are reached, and thereafter decaying with factors $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$.

We suggest here a method for step-size optimisation.

Definition 3.2. *Suppose that we reach $\mathbf{x}^s$ at iteration s. Then α_s^* is* one-step optimal *if it minimises the optimum response function at $\mathbf{x}^{s+1}$. That is, if*

$$\alpha_s^* = \arg \min_{\alpha \in [0,1)} \left\{ \max_{\mathbf{y} \in X} \Psi(\mathbf{x}^{s+1}(\alpha), \mathbf{y}) \right\}. \tag{15}$$

(Recall that $\mathbf{x}^{s+1}$ depends on α.)

This method is intuitively appealing. Indeed, we are trying to force the maximum of the Nikaido-Isoda function that is an equilibrium condition.

Of course, we could also use two-step optimisation, or indeed *n*-step. However any number greater than one would be computationally expensive. We have found that one-step optimising the step sizes leads to fewer iterations, but that each step takes longer to complete than when using constant step sizes.

When we refer to optimal step sizes throughout the paper, we generally mean one-step optimal, however in some cases where optmisation is very simple, other more elementary methods have been used.

4. Some simple examples with static games

In this section we will consider some simple games in order to gain an appreciation for how the algorithm works in conjunction with the Nikaido-Isoda function.

4.1. **Simple two player games.**

4.1.1. *A two player game with identical payoff functions.* Consider a two player game where both players are maximisers, the action space for each player is $\mathbf{R}$ and the payoff functions of players 1 and 2 are identical. Consider the following simple payoff function on the region $X = \{(x_1, x_2) : -10 \leq x_1, x_2 \leq 10\}$.

$$\phi_i(\mathbf{x}) = -x_1^2 - x_2^2 \qquad i = 1, 2. \tag{16}$$

In this case the Nikaido Isoda function is:

$$\Psi(\mathbf{x}, \mathbf{y}) = x_1^2 + x_2^2 - y_1^2 - y_2^2. \tag{17}$$

Notice in this case that assumptions 1-5 of Theorem 3.1 are satisfied.

The Nash equilibrium for this game is clearly the maximum of the common payoff function, that is $\mathbf{x} = (0, 0)$.

The optimum response function $Z(\mathbf{x}) = \arg\max_{\mathbf{y} \in X} \Psi(\mathbf{x}, \mathbf{y})$ is determined by simple calculus to be $(0, 0)$ for any $\mathbf{x}$. Hence if we were to choose α_0 to be ε where $0 < \varepsilon \ll 1$, it doesn't matter what values we assign to $\alpha_1, \alpha_2, \ldots$ since we can choose ε such that $\mathbf{x}^1$ satisfies our termination conditions, namely that the Nikaido-Isoda function

$$\mathbf{x}^1 = \alpha_0 \mathbf{x}^0 + (1 - \alpha_0) Z(\mathbf{x}^0) = \varepsilon \mathbf{x}^0 + (1 - \varepsilon)(0, 0) = \varepsilon \mathbf{x}^0 \simeq (0, 0).$$

The convergence in the action space is displayed in Figure 1 for starting guess $(10, 5)$.

4.1.2. *A slightly less trivial example.* Consider the situation of Section 4.1.1, but let the common payoff function now be

$$\phi_i(\mathbf{x}) = -\frac{(x_1 + x_2)^2}{4} - \frac{(x_1 - x_2)^2}{9}. \tag{18}$$

The Nikaido-Isoda function in this case is

$$\begin{aligned} \Psi(\mathbf{x}, \mathbf{y}) = \quad & 2\left\{ \frac{(x_1 + x_2)^2}{4} + \frac{(x_1 - x_2)^2}{9} \right\} \\ & - \left\{ \frac{(y_1 + x_2)^2}{4} + \frac{(y_1 - x_2)^2}{9} \right\} \\ & - \left\{ \frac{(x_1 + y_2)^2}{4} + \frac{(x_1 - y_2)^2}{9} \right\}. \end{aligned} \tag{19}$$

As in Section 4.1.1, the Nash equilibrium for this game is $\mathbf{x} = (0, 0)$.

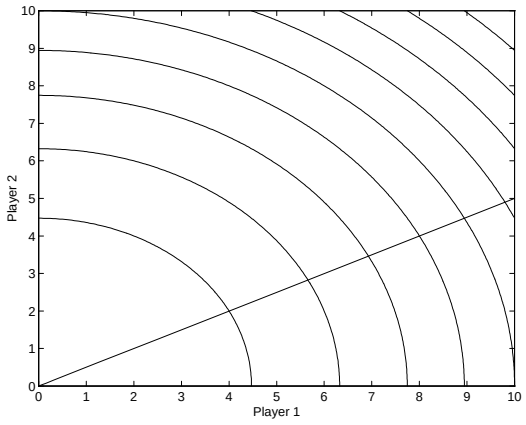

FIGURE 1. Convergence of the example in Section 4.1.1 with starting guess (10,5)

The optimum response function is calculated to be $Z(\mathbf{x}) = -\frac{5}{13}(x_2, x_1)$. In this case it is also relatively simple to see how to optimise the α_s; since both players have the same payoff function, the optimal α_s is the one which optimises the payoff function

$$\phi_i(\mathbf{x^{s+1}}) = \phi_i\left(\alpha_s \mathbf{x^s} + (1-\alpha_s)Z(\mathbf{x^s})\right) \quad 0 < \alpha_s \leq 1. \tag{20}$$

Our Matlab program (see Appendix) gives the results of Table I, with starting guess $\mathbf{x} = (10, 5)$, and using optimised step sizes α_s.

The equilibrium point is still $\mathbf{x} = (0, 0)$, but the algorithm does not converge directly to it. It rather comes in line with the equilibrium first, as seen in Figure 2. We can now make a comparison between the optimised and nonoptimised α_s. The first graph, with the optimisation performed, shows a much quicker convergence. In contrast the second one, which has $\alpha_s \equiv 0.5$ shows a smoother but much slower convergence. The third shows step sizes of $\alpha_s \equiv 1$ (that is, a non-relaxed algorithm.) We would clearly prefer to use optimised step sizes if this could be achieved easily.

Note that Theorem 3.1 gives no guarantee that the algorithm will converge at all in the second or third cases, despite our success.

4.1.3. *The quantity setting duopoly.* In this model, the players' payoff functions are not necessarily the same. Consider a situation where two firms sell an identical product on the same market [5]. Each firm

Iteration (s)	$\mathbf{x_s}$	α_s
0	(10,5)	0.7309
1	(1.2849,-1.4668)	1.0000
2	(0.5639,-0.4942)	1.0000
3	(0.1901,-0.2169)	1.0000
4	(0.0834,-0.0731)	1.0000
5	(0.0281,-0.0321)	1.0000
6	(0.0123,-0.0108)	0.5001
7	(0.0082,-0.0078)	

TABLE I. Convergence of the example in Section 4.1.2

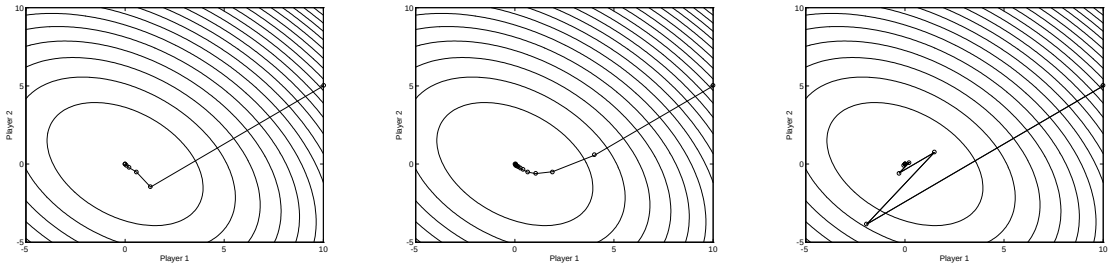

FIGURE 2. Convergence of the example in Section 4.1.2 with starting guess (10,5) using (i) optimised αs, (ii) $\alpha \equiv 0.5$ and (iii) $\alpha \equiv 1$. The isolines are those of the identical payoff function

will want to choose its production rates in such a way as to maximise its respective profits. Let x_i be the production of firm i and let α, λ and ρ be constants. The market price is:

$$p(\mathbf{x}) = \alpha - \rho(x_1 + x_2) \tag{21}$$

and the profit made by firm i, is:

$$\begin{aligned} \phi_i(\mathbf{x}) &= p(\mathbf{x})x_i - \lambda x_i \\ &= [\alpha - \lambda - \rho(x_1 + x_2)]x_i. \end{aligned} \tag{22}$$

The Nikaido-Isoda function in this case is

$$\begin{aligned} \Psi(\mathbf{x},\mathbf{y}) = \quad & [\alpha - \lambda - \rho(y_1 + x_2)]y_1 - [\alpha - \lambda - \rho(x_1 + x_2)]x_1 \\ & + [\alpha - \lambda - \rho(x_1 + y_2)]y_2 - [\alpha - \lambda - \rho(x_1 + x_2)]x_2 \end{aligned} \tag{23}$$

leading to an optimum response function of

$$Z(\mathbf{x}) = \frac{\alpha-\lambda}{2\rho}(1,1) - \frac{1}{2}(x_2, x_1). \tag{24}$$

It is a classical result that the Nash equilibrium in this game is $x_i^N = \frac{\alpha-\lambda}{3\rho}$ with corresponding payoff $\phi_i(x_i^N) = \frac{(\alpha-\lambda)^2}{9\rho}$ (see for example [5]). To show the convergence of the algorithm in this case let us assign values to the parameters. Let $\alpha = 20$, $\lambda = 4$ and $\rho = 1$, then $x^N = \left(\frac{16}{3}, \frac{16}{3}\right)$.

The convergence of the algorithm for the quantity setting duopoly is displayed in Figure 3. Note that $\alpha_s \equiv 0.5$ here.

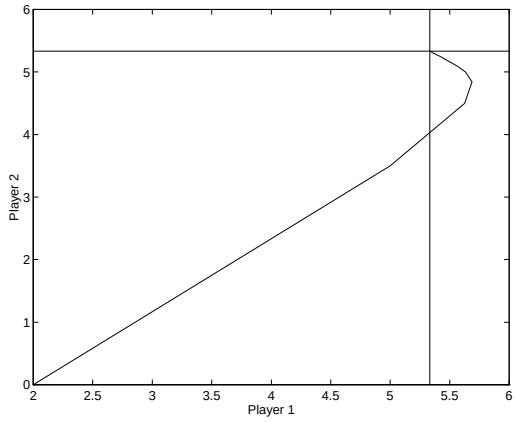

FIGURE 3. Convergence of the example in Section 4.1.3

4.2. **A static game with coupled constraints.** We will now use our method to find the equilibrium of a game with coupled constraints. This means that the players' action set is now a general convex set $X \subset \mathbf{R}^n$ rather than as previously where we were confined to $X = X_1 \times \cdots \times X_n \subset \mathbf{R}^{m_1} \times \cdots \times \mathbf{R}^{m_n}$, each separate X_i being convex.

4.2.1. *Existence of the solution to a coupled constraints game.* The theorems and definitions to follow are presented and proved in Rosen [6], and lead to the existence and uniqueness of an equilibrium in the game we introduce in Section 4.2.2.

Consider the solution to a game where the n players have payoff functions $(\phi_i)_{i=1\ldots n}$,

$$\arg\operatorname{equil}_{\mathbf{x}\in X}\{\phi_1(\mathbf{x}),\ldots,\phi_n(\mathbf{x})\} \tag{25}$$

such that X is a convex subset of $\mathbf{R}^m$. This shall be called a Rosen's coupled constraints game. In the special case where $X = X_1 \times \cdots \times X_n$, the game is said to have uncoupled constraints.

Theorem 4.1. *An equilibrium point exists for every concave n-person game.*

So we know that if each player has a payoff function which is concave with respect to his own action while other players' actions remain fixed, then the game must have at least one Nash equilibrium.

The conditions for uniqueness in the case of uncoupled constraints rely on the concept of diagonal strict concavity. Put loosely, a game with this property is one in which each player has more control over his payoff than the other players have over it.

Definition 4.2. *The function* $f(\mathbf{x},r) = \sum_{i=1}^n r_i\phi_i(\mathbf{x})$ *is called* diagonally strictly concave *for* $\mathbf{x}\in X$ *and fixed* $r \in \boldsymbol{R}_+^n$ *if for every* $\mathbf{x}^0,\mathbf{x}^1 \in X$ *we have*

$$(\mathbf{x}^1-\mathbf{x}^0)'g(\mathbf{x}^0,r)+(\mathbf{x}^0-\mathbf{x}^1)'g(\mathbf{x}^1,r) > 0 \tag{26}$$

where $g(\mathbf{x},r)$ *is the pseudogradient of* f

$$g(\mathbf{x},r) = \begin{bmatrix} r_1\nabla_1\phi_1(\mathbf{x}) \\ \vdots \\ r_n\nabla_n\phi_n(\mathbf{x}) \end{bmatrix}. \tag{27}$$

Theorem 4.3. *In a game with uncoupled constraints, if the joint payoff function* $f(\mathbf{x},r) = \sum_{i=1}^n r_i\phi_i(\mathbf{x})$ *is diagonally strictly concave for some* $r > 0$*, then there exists a unique Nash equilibrium.*

When the constraints are coupled, there are no such guarantees, and we must define a special type of equilibrium.

We introduce the function $h : \mathbf{R}^m \to \mathbf{R}^n$ to describe the constraint set R, where m is the dimension of the collective action space and n is the number of players. Each component of h is a concave function of $\mathbf{x}$, and is defined such that

$$R = \{\mathbf{x} : h(\mathbf{x}) \geq 0\}. \tag{28}$$

Denote the Lagrange multiplier vector for player i by u_i^0, then $\mathbf{x}^0 \in R$ is an equilibrium point if and only if it satisfies the Kuhn-Tucker conditions:

$$h(\mathbf{x}^0) \geq 0 \tag{29}$$

$$(u_i^0)^T h(\mathbf{x}^0) = 0 \tag{30}$$

$$\phi_i(\mathbf{x}^0) \geq \phi_i(y_i|\mathbf{x}^0) + (u_i^0)^T h(y_i|\mathbf{x}^0). \tag{31}$$

Definition 4.4. *The equilibrium point* $\mathbf{x}^0$ *is a* Rosen normalised equilibrium point *if, for some vector* $r > 0$ *and constant* $u^0 \geq 0$*,*

$$u_i^0 = \frac{u^0}{r_i} \tag{32}$$

for each i.

Theorem 4.5. *There exists a normalised equilibrium point to a concave n-person game for every* $r > 0$*.*

Theorem 4.6. *Let Q be a convex subset of* $\boldsymbol{R}_+^n$*. If* $f(\mathbf{x}, r)$ *is diagonally strictly concave for every* $r \in Q$*, then for each* $r \in Q$ *there is a unique normalised equilibrium point.*

This tells us that, if we can show that our game is diagonally strictly concave, then we can find a unique normalised equilibrium.

We will not prove the applicability of Theorem 3.1 to the coupled constraints game. We suggest this could be done if the components of the Nikaido-Isoda function were the players' Lagrangians rather than their payoffs ϕ_i. Our numerical experiments do seem to indicate that the diagonal strict concavity, which guarantees a unique solution to this class of games, is a sufficient condition for the relaxation algorithm to find an equilibrium here.

4.2.2. *Formulation of a coupled constraints game. River Basin Pollution,* [3]. In this game, we consider three players $j = 1, 2, 3$ located along a river. Each agent is engaged in an economic activity at a chosen level x_j, but the players must meet environmental conditions set by a local authority.

Pollutants may be expelled into the river, where they disperse. Two monitoring stations $\ell = 1, 2$ are located along the river, at which the local authority has set maximum pollutant concentration levels.

The revenue for player j is

$$R_j(\mathbf{x}) = [d_1 - d_2(x_1 + x_2 + x_3)]x_j \tag{33}$$

with expenditure

$$F_j(\mathbf{x}) = (c_{1j} + c_{2j}x_j)x_j. \tag{34}$$

Thus the net profit for player j is

$$\begin{aligned} \phi_j(\mathbf{x}) &= R_j(\mathbf{x}) - F_j(\mathbf{x}) \\ &= [d_1 - d_2(x_1 + x_2 + x_3) - c_{1j} - c_{2j}x_j]x_j. \end{aligned} \tag{35}$$

The constraint imposed by the local authority at location l is

$$q_\ell(\mathbf{x}) = \sum_{j=1}^{3} \alpha_{j\ell} e_j x_j \leq 100, \ \ell = 1, 2. \tag{36}$$

The economic constants d_1 and d_2 determine the inverse demand law and are set to 3.0 and 0.01 respectively. The values for constants c_{1j} and c_{2j} are given values in Table II.

The $\alpha_{j\ell}$ are the decay-transportation coefficients from player j to location l, and e_j is the emission coefficient of player j, also given in Table II.

Player j	c_{1j}	c_{2j}	e_j	α_{j1}	α_{j2}
1	0.10	0.01	0.50	6.5	4.583
2	0.12	0.05	0.25	5.0	6.250
3	0.15	0.01	0.75	5.5	3.750

TABLE II. Constants for the River Basin Pollution game

Iteration (s)	x_1^s	x_2^s	x_3^s	α_s
0	0	0	0	0.5
1	9.68	8.59	1.90	0.5
2	14.85	12.62	2.655	0.5
3	17.65	14.49	2.913	0.5
4	19.18	15.35	2.961	0.5
5	20.03	15.73	2.934	0.5
⋮	⋮	⋮	⋮	⋮
10	21.07	16.03	2.762	0.5
⋮	⋮	⋮	⋮	⋮
20	21.14	16.03	2.728	0.5

TABLE III. Convergence in the river basin pollution game

In this case, we can check conditions (26) to show that problem (25) is a diagonally strictly concave one, and so a unique equilibrium point exists. We can thus compute it for a given set of weights r using our relaxation algorithm, where the optimum response function $Z(\mathbf{x}^s)$ is restricted to lie within the feasibility set R. Notice also that the region defined by Equation (36) is convex.

4.2.3. *Solution to the River Basin Pollution game.* Here, we build the Nikaido-Isoda function. According to Section 4.2.1, its components should be the players' corresponding Lagrangian payoffs. However, we will formulate the Nikaido-Isoda function from the unconstrained game, letting the constraints be handled within the Matlab procedure.

To this end, the Nikaido-Isoda function is

$$\begin{aligned}\Psi(\mathbf{x},(\mathbf{y})) &= \sum_{j=1}^{3}(\phi_j(y_i|\mathbf{x})-\phi_j(\mathbf{x}))\\ &= [d_1-d_2(y_1+x_2+x_3)-c_{11}-c_{21}y_1]y_1\\ &\quad +[d_1-d_2(x_1+y_2+x_3)-c_{12}-c_{22}y_2]y_2\\ &\quad +[d_1-d_2(x_1+x_2+y_3)-c_{13}-c_{23}y_3]y_3.\end{aligned}$$

We used a starting guess of $\mathbf{x}=(0,0,0)$ and $\alpha\equiv 0.5$ in our Matlab program. The convergence is shown in Table III.

The convergence is displayed as a line in the 3D action space in Figure 4. This game was also solved in [3] using Rosen's algorithm, and was found to have equilibrium $\mathbf{x} = (21.149, 16.028, 2.722)$, giving net profits $\mathbf{z} = (48.42, 26.92, 6.60)$. The first constraint is active, *i.e.* $q_1(\mathbf{x}) = 100$; the second constraint is inactive ($q_2(\mathbf{x}) = 81.17$). Our solution of $\mathbf{x} = (21.14, 16.03, 2.728)$ is within ± 0.01 of the solution achieved in [3].

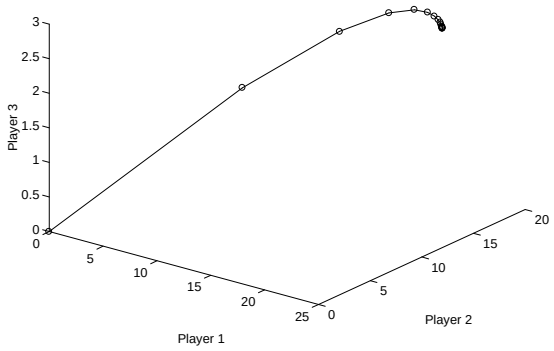

FIGURE 4. Convergence in Section 4.2.2 with starting guess (0,0,0), $\alpha \equiv 0.5$

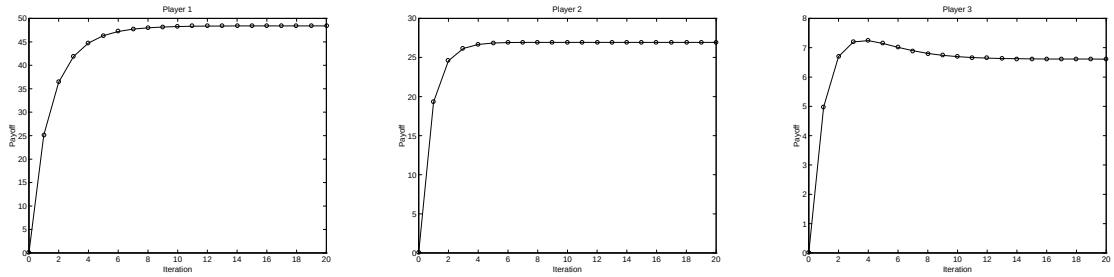

FIGURE 5. Progression of payoffs of Figure 4

4.2.4. *Applying the Pigouvian taxes.* Now that the Nash normalised equilibrium has been found, we can force the players to obey it by applying Pigouvian taxes. In this way we create a new, unconstrained game whose equilibrium is the same as that of the original.

For each active constraint, we place a tax on each player of the amount

$$T_\ell(\mathbf{x}) = \max(0, \lambda_\ell g_\ell(\mathbf{x})) \tag{37}$$

where λ_ℓ is the Lagrange multiplier for constraint ℓ at the equilibrium $\mathbf{x} = \mathbf{x}^*$.

For our game, recall that $g_\ell = q_\ell$ is the constraint at location ℓ, $\ell = 1, 2$, and that the only the first constraint is active.

Thus, in the river basin pollution game with the parameters given, the payoff function for player j becomes

$$\begin{aligned} \phi_j^*(\mathbf{x}) &= R_j(\mathbf{x}) - F_j(\mathbf{x}) - \sum_\ell T_\ell(\mathbf{x}) \\ &= [d_1 - d_2(x_1 + x_2 + x_3) - c_{1j} - c_{2j}x_j]x_j - \lambda_1 \max\left(0, \sum_{j=1}^{3} \alpha_{j\ell} e_j x_j - 100\right). \end{aligned} \tag{38}$$

The coputations of Section 4.2.3 gave us the Lagrange multiplier value for the active constraint $\lambda_1 = 0.5774$. Thus, applying the taxes as above leads to the set of payoff functions ϕ_j^* described by Equation (38), which has the unconstrained Nash equilibrium $\mathbf{x}^{**} = \arg\text{equil}_{\mathbf{x}\in\mathbb{R}^m}(\phi_1, \phi_2, \phi_3, q_1, q_2) = \mathbf{x}^*$. That is, the new (unconstrained) Nash equilibrium is equal to the old (constrained) Nash normalised equilibrium.

We report that checking numerically the validity of the above is difficult (using the Matlab `constr` routine, see Appendix) due to the derivative discontinuities introduced by the max function. However, cross-sectional graphs of the payoff functions strongly vindicate the results (see Figure 6).

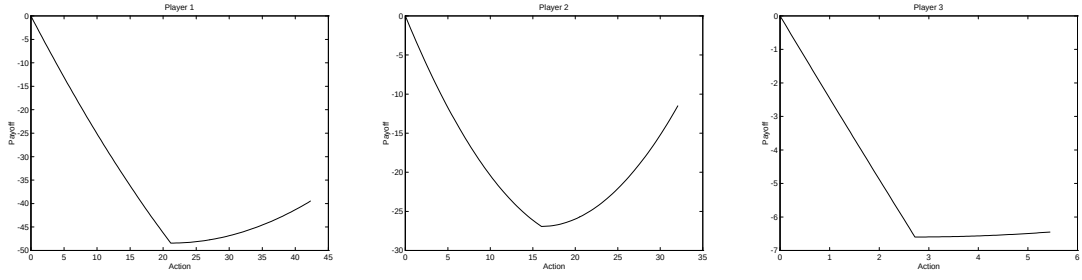

FIGURE 6. Payoff functions for players 1,2,3 with Pigouvian taxes applied

5. FINDING EQUILIBRIA IN DYNAMIC GAMES

5.1. **Application to dynamic games.** The relaxation algorithm allows us to find solutions to a large number of dynamic games with coupled constraints, so long as those games can be decomposed to a game where the present value of the payoff for each player is a function of the future collective actions and the initial conditions of the game. This is the case for deterministic games with the open-loop information pattern.

In particular, any finite horizon game where the coupled constraints and payoffs at time t depend only on the players' states at that time, and the states can be chosen by players through their actions, is solvable by this method.

Let us illustrate this with an example where the game of Section 4.2.2 is played over a finite number of periods.

5.2. **Formulation of the game.** *The River Basin Pollution game played over T periods.* Suppose that the agents of Section 4.2.2 now repeat the River Pollution game over T discrete time periods. They have an option to invest in their products and capacities, which can depreciate.

The values for all constants will remain the same as in Section 4.2.2 over the period of the game.

State equations. Let $x_i^{(t)}$ denote the fully utilised production capacity that player i has at time t, governed by the state equation

$$x_i^{(t+1)} = (1-\mu_i)x_i^{(t)} + u_i^{(t)} \tag{39}$$

where $u_i^{(t)}$ is the investment that player i makes at time t, and μ_i is the depreciation that applies to player i, and $x_i^{(0)}$ is fixed, $i = 1,2,3$, $t = 0,\dots,T$.

Denote the collective action and state respectively by

$$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \qquad \text{and} \tag{40}$$

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \tag{41}$$

where $\mathbf{u}_i = (u_i^{(1)},\dots,u_i^{(T)})$ is the vector of all actions of player i and $\mathbf{x}_i = (x_i^{(1)},\dots,x_i^{(T)})$ is the collective-time state vector of player i. Denote the collective action and state at time t by

$$\mathbf{u}^{(t)} = \left(u_1^{(t)}, u_2^{(t)}, u_3^{(t)}\right) \qquad \text{and} \tag{42}$$

$$\mathbf{x}^{(t)} = \left(x_1^{(t)}, x_2^{(t)}, x_3^{(t)}\right) \tag{43}$$

respectively.

Information structure. Suppose that the game is open-loop. That is, the players cannot observe the state or capacity of another player other than at $t = 0$. This implies that the game can have an open-loop Nash equilibrium[2].

[2]We consider this solution concept valid for "short" horizon dynamic games, where the players are in the process of learning each others' behaviour. In that case, the expression $\rho^T k x_i^{(T)}$ in Equation (44) can represent the payoff in a game which is played from T onwards, under a different information pattern.

Payoff function. The present value payoff for player i is

$$\Phi_i(\mathbf{u}) = \sum_{t=0}^{T-1} \rho^t \left(\phi_i(\mathbf{x}^{(t)}) - c_{3i}(u_i^{(t)})^2 \right) + \rho^T k_i x_i^{(T)} \qquad i = 1,2,3 \tag{44}$$

where $\phi_i(\mathbf{x}^{(t)})$ is defined through Equation (35), $c_{3i}(u_i^{(t)})^2$ is the investment cost at time t and $\rho^T k_i x_i^{(T)}$ represents the scrap value and time T for player i, $k_i \geq 0$, being the scrap value constant for player i.

Constraints. The constraints of Equation (36) for $\ell = 1,2$ are to be satisfied by the $\mathbf{x}^{(t)}$ for all $t \in \{0,1,\ldots,T\}$. The current production capacity $x_i^{(t)}$ must be nonnegative, but the $u_i^{(t)}$ may be negative, in which case a withdrawal of capital would be indicated.

5.3. **Solution to the game.** We want to find the normalised Nash equilibrium

$$\arg\operatorname{equil}_{\mathbf{u}\in\mathbf{U}} \{\Phi_1, \Phi_2, \Phi_3\} \tag{45}$$

subject to the state equations and constraints given above.

To do this using the relaxation algorithm, it is necessary to reformulate the payoff functions and constraints so that they contain only constants and the actions. That is, they need to be free of the states for us to start.

The reformulation for the payoff functions is

$$\begin{aligned} \Phi_i(\mathbf{u}) \quad = \quad & \sum_{t=0}^{T-1} \rho^t \left\{ \phi_i \left(\begin{bmatrix} (1-\mu_1)x_1^{(0)} + \sum_{s=1}^{t}(1-\mu_1)^{t-s}u_1^{(s)} \\ (1-\mu_2)x_2^{(0)} + \sum_{s=1}^{t}(1-\mu_2)^{t-s}u_2^{(s)} \\ (1-\mu_3)x_3^{(0)} + \sum_{s=1}^{t}(1-\mu_3)^{t-s}u_3^{(s)} \end{bmatrix} \right) - c_{3i}(u_i^{(t)})^2 \right\} \\ & + \rho^T \left\{ (1-\mu_i)x_i^{(0)} + \sum_{s=1}^{T}(1-\mu_i)^{t-s}u_i^{(s)} \right\} \qquad i = 1,2,3 \end{aligned} \tag{46}$$

where the sums over s are interpreted as zero if t or T are respectively zero. The constraints become

$$q_\ell(\mathbf{u}^{(t)}) = \sum_{i=1}^{3} \alpha_{j\ell} e_j \left((1-\mu_i)x_i^{(0)} + \sum_{s=1}^{t}(1-\mu_i)^{t-s}u_i^{(s)} \right) \leq 100 \tag{47}$$

for $\ell = 1,2$ and $t \in \{0,1,\ldots T-1\}$, where the sum over s is again treated as zero if $t = 0$.

In the rest of this section, we will be concerned with the players' first period actions, *i.e.* $u_1^{(0)}$. The second period action depends on the parameters as follows:

$$u_i^{(1)} = \frac{\rho k}{2c_{3i}}. \tag{48}$$

We now solve the above game for some different parameter combinations.

Two periods with no depreciation or scrap value. Let $T = 2$, $\rho \in (0,1]$, $\mu_i \equiv 0$, $k_i \equiv 0$, $c_{3i} \equiv 1$ and let the initial state vector be the Nash solution of the game over one period, namely $\mathbf{x}^{(0)} = (21.149, 16.030, 2.722)$.

Given this information, and a starting guess of $\mathbf{u} = ((0,0),(0,0),(0,0))$, which is the expected outcome, the program output is the starting guess.

This is intuitively correct since we are just playing an identical game twice, with a penalty but no reward associated with investment, and thus expect the Nash solution for the single period to be repeated over both periods. The constraints' satisfaction in each period is the same as in the static game.

Two periods with depreciation but no discounting or scrap value. Let $T = 2$, $\rho = 1$, $\mu_i \equiv 0.1$, $k_i \equiv 0$, $c_{3i} \equiv 1$ and let the initial state vector be the Nash solution of the game over one period, namely $\mathbf{x}^{(0)} = (21.149, 16.030, 2.722)$.

The solution in this case, with the same starting guess as before, is

$$\mathbf{u} = ((0.9577, 0), (0.4305, 0), (1.1782, 0)).$$

That is, the players adjust the states to $\mathbf{x}^{(1)} = (19.992, 14.858, 3.628)$ (in period 1), and make no further investment. The payoffs corresponding to the above actions are $(93.79, 52.77, 14.02)$. The fact that the state does not revert to the initial state is caused by the presence of investment costs. The constraints for period 1 are inactive ($q_1 = -1.49$, $q_2 = -20.77$.)

Two periods with scrap value and discounting but no depreciation. Let $T = 2$, $\mu_i \equiv 1$, $c_{3i} \equiv 1$, $k = 1$ and $\mathbf{x}^{(0)} = (21.149, 16.030, 2.722)$ as before, but now let ρ be less than one.

With starting guess $\mathbf{u} = ((0,0),(0,0),(0,0))$, ρ was varied with the resulting payoff evolution shown in Figure 7. As in one agent problems, the larger ρ the higher the payoffs. The payoffs are achieved through

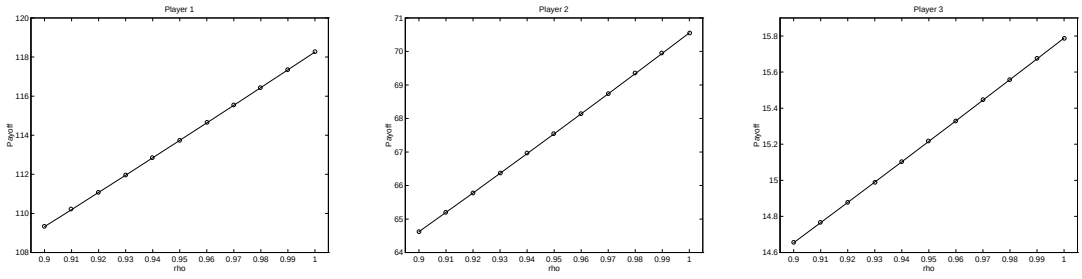

FIGURE 7. How payoff varies with ρ for players 1,2,3 respectively

actions shown in Table IV and resulting in the state evolution as in Figure 8. Figure 9 compares the first period actions of the players (*i.e.* $u_i^{(1)}$, $i = 1, 2, 3$) for different discounting factors.

ρ	$\mathbf{u}_1$		$\mathbf{u}_2$		$\mathbf{u}_3$	
1	0.0203	0.500	0.2985	0.500	-0.1064	0.500
0.99	0.0199	0.495	0.2927	0.495	-0.1044	0.495
0.98	0.0195	0.490	0.2869	0.490	-0.1023	0.490
0.97	0.0191	0.485	0.2812	0.485	-0.1002	0.485

TABLE IV. Nash equilibria for varying ρ in Section 5.2

For the static equilibrium of Section 4.2.2 as well as for all dynamic equilibria computed in this section, the first of the constraints ($\ell = 1$) has been active. An increase in production by players 1 and 2 is

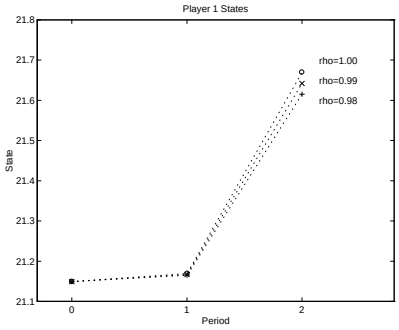

FIGURE 8. Player 1 states for varying ρ

compensated by the production decrease of player 3 who is the heaviest polluter and the highest cost producer (see Table II.)

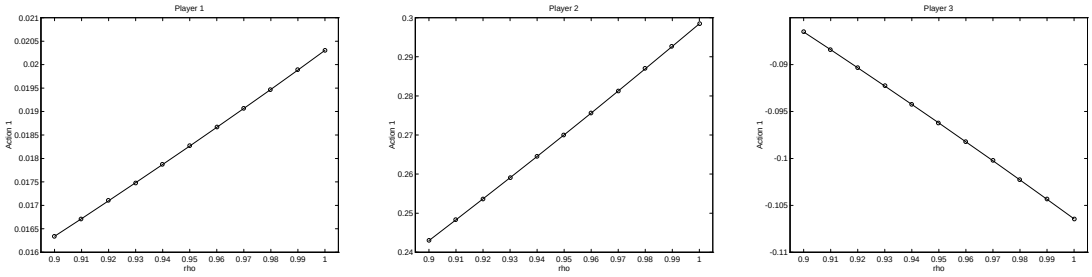

FIGURE 9. Player 1, 2 and 3 actions ($t = 0$) for varying ρ

Two periods with barely active constraints. We look here at how the Nash equilibrium behaves when a constraint is non-active in the first period ($t = 0$). In this way we can observe the behaviour of the players when constraints become active.

Redefine the constraints as

$$q_1(\mathbf{u}^{(t)}) = \sum_{i=1}^{3} \alpha_{j1} e_j \left((1-\mu_i) x_i^{(0)} + \sum_{s=1}^{t} (1-\mu_i)^{t-s} u_i^{(s)} \right) \leq 424 \tag{49}$$

$$q_2(\mathbf{u}^{(t)}) = \sum_{i=1}^{3} \alpha_{j2} e_j \left((1-\mu_i) x_i^{(0)} + \sum_{s=1}^{t} (1-\mu_i)^{t-s} u_i^{(s)} \right) \leq 304, \tag{50}$$

then the constraints are nonactive when the game is played for a single period. However, when we play for two periods and vary ρ, the constraints become active.

Firstly, setting μ_j to zero for $j = 1, 2, 3$, and letting ρ vary from 0.90 to 1.00 give the payoffs and actions of Figures 10 and 12 respectively.

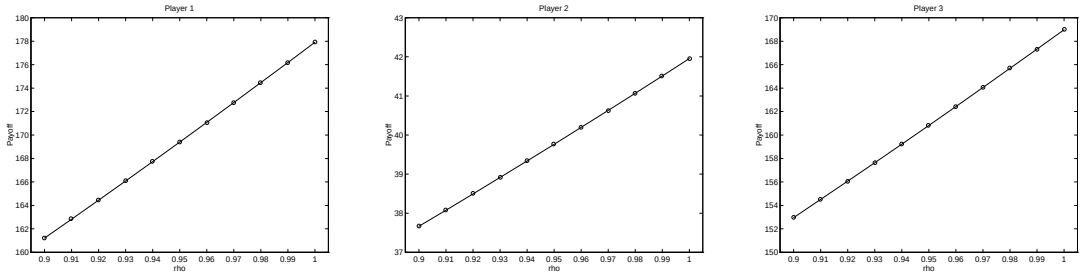

FIGURE 10. How payoff varies with ρ for players 1,2,3 respectively for the new constraints

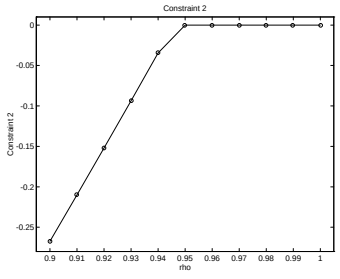

FIGURE 11. How constraint values vary with ρ for the new constraints

Again, the payoffs are high when the discount factor is large. Regarding the players' actions shown in Figure 12, they are such that for a certain value of ρ (here, $\rho = .95$ see Figure 11) the second constraint becomes binding. From this value of ρ "onwards" the least economic agent's actions (*i.e.* Player 3's; see his parameteres in Table II) start diminishing.

The evolution of the state is shown for Player 1 in Figure 13. We note that his capacities in period 1 surpass those of the "static" equilibrium, and grow in the discount factor. An apparently big jump in capacities in period 2 is mainly due to the relatively large weight of the scrap value in the agent's objective function.

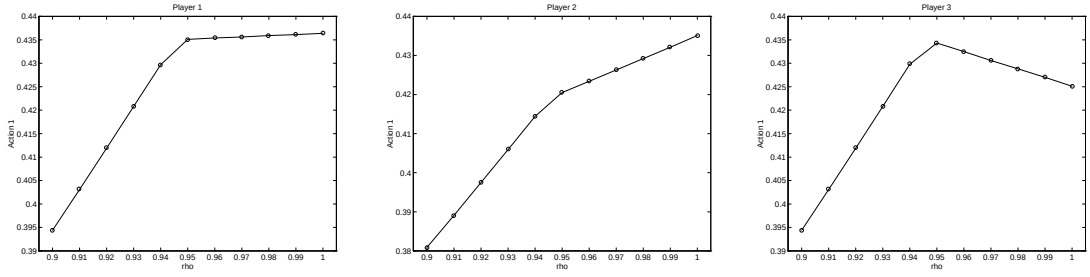

FIGURE 12. Player 1, 2 and 3 actions ($t = 0$) for varying ρ with the new constraints

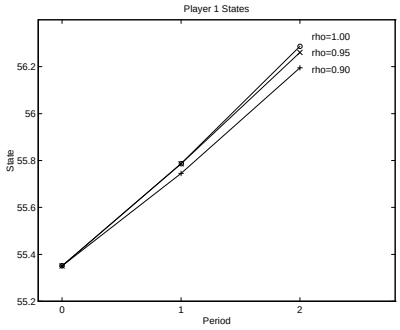

FIGURE 13. Player 1 states for varying ρ with the new constraints

Secondly, varying μ_j from being 0 to 0.10 (for all players simultaneously) and keeping $\rho = 0.96$, we get the results displayed in Figures 14, 15 and 16.

The highest payoffs are (obviously) for no depreciation ($\mu = 0$). The impact of contraints on the equilibrium actions is noticeable in this case too. For the depreciation parameter values close to zero, a contraint becomes binding and the players have to diminish their actions.

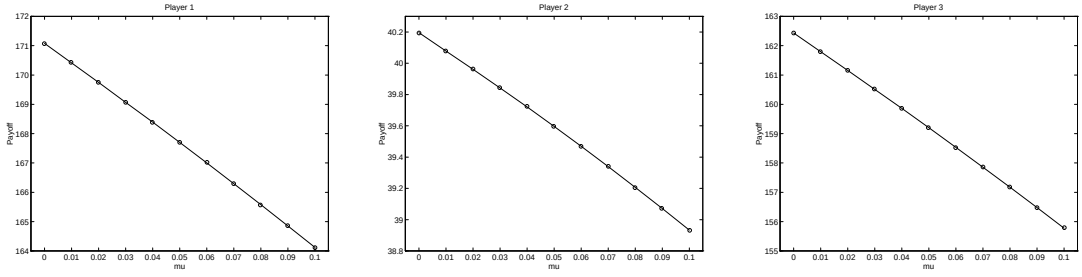

FIGURE 14. Player 1, 2 and 3 payoffs for varying μ_j with the new constraints

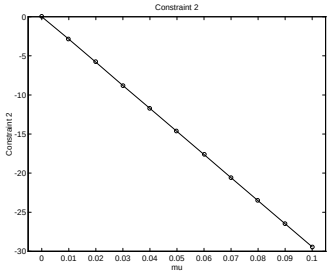

FIGURE 15. How constraint values vary with μ_j for the new constraints

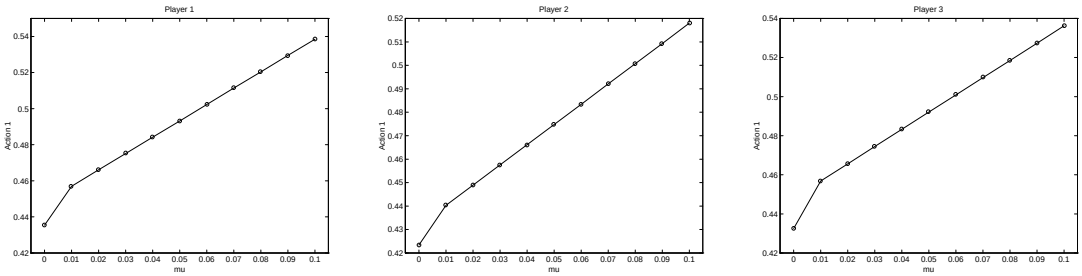

FIGURE 16. Player 1, 2 and 3 actions for varying μ_j with the new constraints

6. CONCLUSIONS

The Nikaido-Isoda function was introduced as a means of making the problem of finding a Nash equilibrium one of maximising a multivariate function.

This together with the relaxation methodology allows us to find the Nash equilibrium of many non-zero sum multiplayer games in a much simpler fashion than has previously been possible. Software was developed to implement these ideas and has been used successfully to solve some nontrivial examples.

The games we solved included several elementary static games and some less elementary games including the dynamic river basin pollution game based on [3]. The latter was originally solved using Rosen's Algorithm.

The developed software enabled us to find the Nash normalised equilibrium of a two period dynamic game played under the open loop information pattern. In particular, we observed how varying the discount factor and capital depreciation influence the equilibrium.

We noted that if a one period game has a (static) Rosen normalised equilibrium that makes a constraint active, this constraint usually remains active in the dynamic game. The players' investment behaviour is such that the heaviest polluter should decrease production for the game to have an equilibrium. All players' payoffs decrease with a diminishing discount factor.

However, if a static game has an unconstrained static equilibrium, the dynamic equilibrium generally implies an increased production by all players and that (usually one of) the constraints becomes active. Also, the more important the future becomes (*i.e.* ρ approaches 1) the "greedier" the players become and the more likely is an equilibrium with active constraints. In real life this may mean that players should be more strictly monitored for larger values of ρ. We summarise that in dynamic games with coupled constraints and a final payoff function, the equilibrium (45), with $\rho \to 1$, moves *away* from the static, or "repeated", equilibrium of Section 4.2.3.

Notice that all dynamic equilibria are Rosen normalised. In order to force the players to invest according to these equilibria, Pigouvian taxes can be imposed [3].

The relaxation methodology increases the ability of game theory to solve ever more complex problems, and does so using simple ideas.

APPENDIX A. NOTES ON CUSTOMARILY DESIGNED MATLAB SOFTWARE FOR SOLUTIONS OF GAMES

This section contains a brief description of the MATLAB program we designed to implement the relaxation algorithm.

A.1. **Program design.** The flow diagram is displayed in Figure 17.

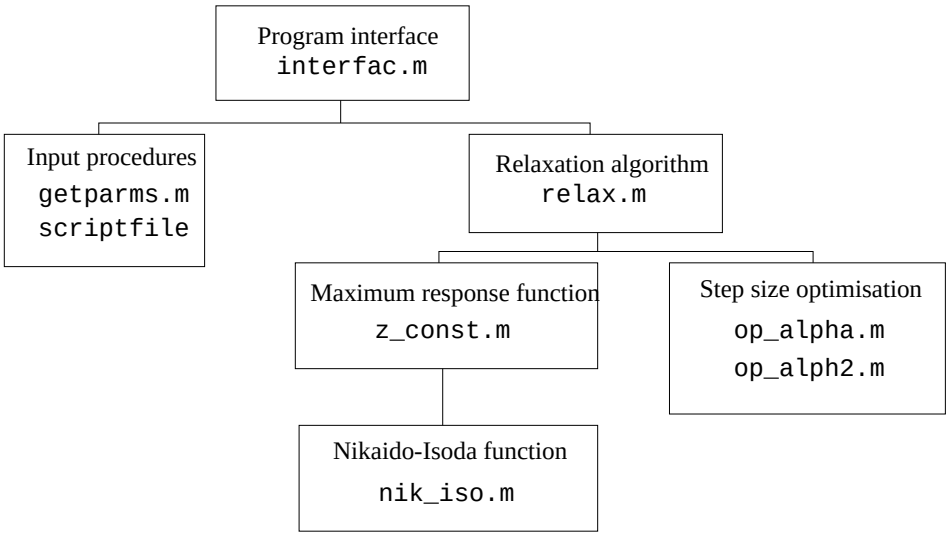

FIGURE 17. Program flow diagram

A.2. **Description of the procedures.**

A.2.1. *User interface* `interfac.m`. This reads in required parameters which describe the game to be solved, and feeds them to the algorithm. It then outputs the results to screen.

A.2.2. *Parameter input* `getparms.m, scriptfile`. The input to the program can be either through a series of prompted questions or through use of a MATLAB m-file script which takes no arguments and returns all required parameters to the program.

The required parameters at the time of writing are the names of the payoff function and constraint function, a description of the game, a vector giving the dimension of each player's action space, the lower and upper bounds on the players' actions, the number of equality constraints, the precision required, the step-size optimisation method and a starting guess.

A.2.3. *The payoff function.* This takes the collective action and the player number and returns the corresponding payoff.

A.2.4. *The constraint function.* This takes the collective action vector and returns a vector of numbers which must be nonpositive to satisfy the constraints. Some may be required to be equal to zero if the number of equality constraints is positive.

A.2.5. *The relaxation algorithm* `relax.m`. The main procedure containing the control loop for the algorithm. This function takes the required parameters, and iterates the algorithm until the termination conditions have been reached. It returns the normalised Nash equilibrium together with the points traversed in achieving the solution, the individual payoffs at each step, the step sizes used and the time taken.

A.2.6. *Next iterate* `nextx.m`. This simple function takes the x^s, $Z(x^s)$ and α and returns x^{s+1}.

A.2.7. *Nikaido-Isoda function* `nik_iso.m`. Returns the value of the Nikaido-Isoda function at the point $(\mathbf{y},\mathbf{x})$. The order of the arguments is the reverse of that in the paper for technical reasons.

A.2.8. *Optimum response function* `z_const.m`. This returns the $\mathbf{y}$ which maximises the Nikaido-Isoda function at the point $\mathbf{x}$.

A.2.9. *Step-size optimisation primitive method* `op_alpha.m`. This simple step-size optimisation only works when the players have identical payoff functions. It simply determines the α which maximises the common payoff.

A.2.10. *Step-size optimisation general method* `op_alph2.m`. This step-size optimisation works by trying to minimise the maximum of the Nikaido-Isoda function at the next step.

A.2.11. *Vector to string function* `vect2str.m`. Takes a collective action and the dimensions of the players' action spaces, and returns a string output of a nested vector which can be output to screen. For example, in the dynamic river pollution game, where each player had two actions, the solution was $[[0,0][0,0][0,0]]$ for $\rho = 1$.

REFERENCES

[1] J.P. AUBIN, 1979, "Mathematical Methods of Game and Economic Theory" *Amsterdam, North Holland.*

[2] T. BAŞAR 1987, "Relaxation techniques and asynchronous algorithms for online computation of noncooperative equilibria", *Journal of Economic Dynamics and Control*, Vol. 11, pp. 531-549.

[3] ALAIN HAURIE & JACEK B. KRAWCZYK, 1997, "Optimal charges on river effluent from lumped and distributed sources", *Environmental Modelling and Assessment*, vol. 2, no.3, pp 93-106.

[4] NIKAIDO HUKUHANE AND ISODA KAZUO, 1955, "Note on noncooperative convex games", *Pacific Journal of Mathematics*, Vol. 5, Supp. 1, pp 807-815.

[5] JACEK B. KRAWCZYK, 1995, "ECON 407 - Economic Dynamics B, Introduction to Dynamic Games with Application - Course Notes", *Printing Unit, Works and Services, Victoria University of Wellington* 1774/95/5.

[6] J. B. ROSEN, 1965, "Existence and uniqueness of equilibrium points for concave n-person games", *Econometrica*, Vol. 33, No. 3, pp 520-534.

[7] S. P. URYAS'EV, 1990, -"Адаптивные Алгоритмы Стохастическоj0 Оптимизации и Теории Игр " ("Adaptive Algorithms for Stochastic Optimisation and Game Theory,") Москва «Наука».

[8] STANISLAV URYAS'EV & REUVEN Y. RUBINSTEIN, 1994, "On Relaxation Algorithms in Computation of Noncooperative Equilibria", *IEEE Transactions on Automatic Control*, Vol. 39, No. 6. pp. 1263-1267.