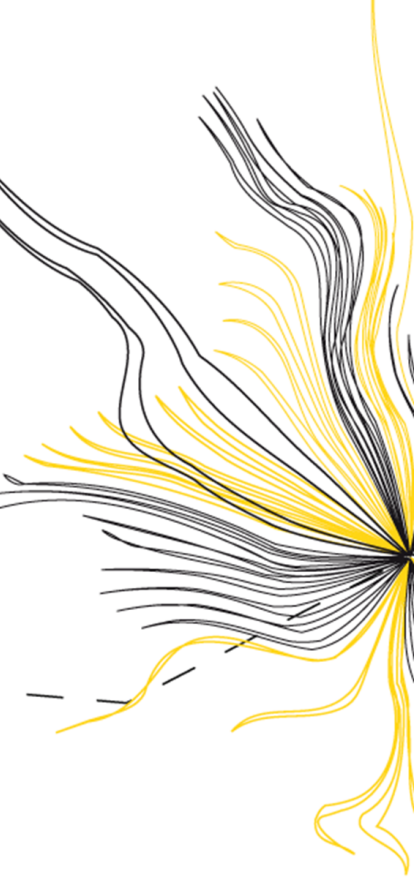


UNIVERSITY OF TWENTE.

Faculty of Behavioural Management & Social Sciences

M.Sc. Program: Industrial Engineering & Management

Specialization: *Financial Engineering & Management*

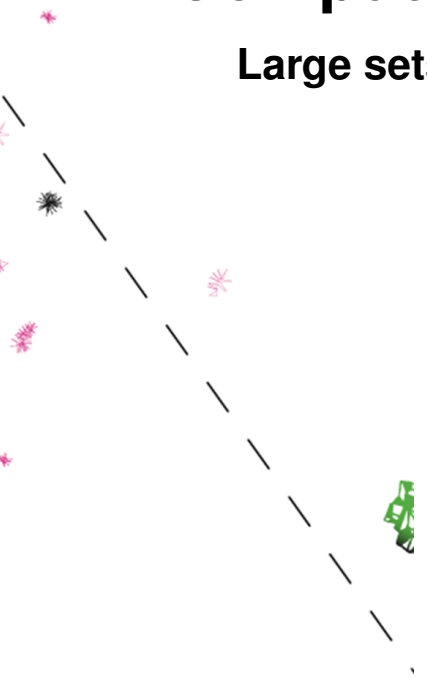
Master Thesis

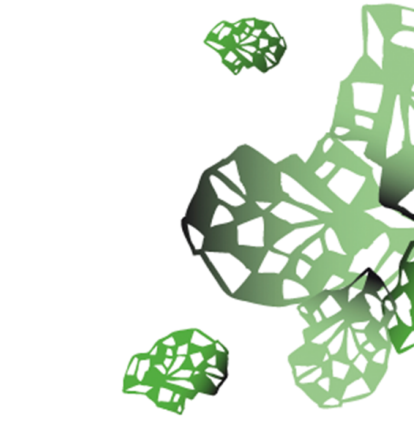
Computations in Stochastic Game Theory

Large sets of rewards in games with communicating states and frequency-dependent transition probabilities and stage payoffs

Llea Nasira Samuel
s1543261

April 28, 2017





Supervisor:
dr. R. A. M. G. Joosten

Industrial Engineering & Business Information Systems
Faculty of Behavioural, Management and Social Sciences
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Master Thesis, University of Twente.

Computations in Stochastic Game Theory. Large sets of rewards in games with communicating states and frequency-dependent transition probabilities and stage payoffs.

This research project was performed under the supervision of dr. R. A. M. G. Joosten in partial fulfillment of the Master of Science degree in Industrial Engineering & Management.

Faulty of Behavioural Management & Social Sciences (BMS)

April 28, 2017

Acknowledgements

I am of the firm belief that in order to arrive at the stage of producing a Master thesis of a level that has resulted in the ability to publish new findings, had been due to the efforts of many more individuals than just the author of this thesis.

First and most obvious, Reinoud Joosten. Without him, I would still be floundering without an idea for a thesis project, much less the invaluable assistance he has provided during the project's development. His unique viewpoint into stochastic game theory and ability to accurately explain the crossover from theoretic to computational has proven to be irreplaceable.

Also to Dr. Berend Roorda, whose insight saved my thesis from being a collection of technical jargon instead of the easy-to-follow read-through most readers will experience in the coming pages.

Second, my family and close friends to whom I owe the mental fortitude that allows me to step confidently and happily from one day into the next.

Thirdly, the administration at the University of Twente, specifically, the International Office, Student Services and those overseeing us Industrial Engineering & Management M.Sc. students.

To anyone else I may have forgotten, you know who you are. Thank you.

Now...on with the show.

Abstract

With the amount of conceptual literature on 2-person social dilemmas, this thesis takes the necessary step of creating an algorithm that is able to compute what has been modeled. That is: dynamic 2-state, 2-player, 2-action stochastic competitive games with allowances for frequency-dependent transition probabilities and frequency-dependent stage payoffs. To demonstrate its usage, I employ a '*Commons*'-type social dilemma where 2 players compete over a *renewable common-pool* resource. In this example, both the transition probabilities and stage payoffs possess frequency-dependencies that are linear. Note that adherence to such a limitation is not necessary for the stage payoffs, but may be for the transition probabilities.

Results in both the repeated game model and the two stochastic models are in line with results found in prior studies. That is, frequency-dependent stage payoffs cause a reduction in the value of the payoffs available as the reward set heads south-west in $\mathbb{R}^2$. More noteworthy, results in the stochastic models with frequency-dependent stage payoffs show that frequency-dependent transition probabilities have significant *reduction effects* on the *probability* of obtaining a higher payoff for either player. The reward sets in both situations increase in size by stretching toward the origin, compared to the frequency-independent stage payoffs reward sets. This increase is compensated in part through a narrowing of the set's shape in $\mathbb{R}^2$ space which results in smaller differences between both players when one gains more than the other.

Keywords Repeated Games, Stochastic Games, Limiting Average Rewards, Frequency-Dependent Transition Probabilities, Frequency-Dependent Stage Payoffs

Contents

List of Figures

1. Introduction

Van Lange, Joireman, Parks and Van Dijk (2013) define social dilemmas as

> *...situations in which a non-cooperative course of action is (at times) tempting for each individual in that it yields superior (often short-term) outcomes for self, and if all pursue this non-cooperative course of action, all are (often in the longer-term) worse off than if all had cooperated.*

From such a definition, one can conclude that social dilemmas can be either momentary or dynamic. The dynamic element contains both short term and long term outcomes.

To better understand the significance of these differentiations, I will explain one of the most common examples of a social dilemma: The Prisoner's Dilemma. In this scenario, two prisoners must decide whether or not to rat each other out to the police in order to receive a reduced sentence handed down by the judicial system. More specifically, each prisoner (arbitrary labels 'A' and 'B') is aware that if A rats out B, A receives a reduced sentence for his (her) cooperation with the judicial system, while B receives the largest sentence possible. If both A and B rat each other out, reduced sentences are give to each. But such a reduction is trivial when compared to the time added on due to the other's betrayal.

In game theoretic terms this reduced sentenced is called the *payoff* and the choices of whether to cooperate with the police or not are the *actions*. The prisoners (usually two, but can be more) are called *players*. The entire situation is referred to as a *game*.

For the Prisoner's Dilemma game, each player has a dominant action: to rat the other out.

Because if one rats the other out, whilst the other does not, the *defector* gets a significantly reduced sentence. But if they both defect, they both get harsh sentences. This pair of harsh sentences is the Nash equilibrium as they correspond to each player's dominant action choice. However, there is also the option of their both remaining silent. This gives a smaller sentence to each which is not the dominant action, but it is still better than the Nash equilibrium. This idea is considered through *Pareto optimality*. Hans Peters (2008) defines:

> *A pair of strategies is Pareto optimal if there is no other pair such that the associated payoffs are at least as good for both players and strictly better for at least one player.*

This makes the Nash equilibrium *Pareto inferior*, and the choice of remaining silent is *Pareto superior*.

When considering the time element, first assume that both prisoners cooperate and remain silent. If this is a finitely repeating game (say the situation happens 5 times), then on the fifth play, both players will feel inclined to rat the other out, as the situation will never again repeat itself thus incurring the wrath of the other. So for the fifth iteration, the dominant action is played. However, as both players suspect the other of this at the fourth repetition, they use this earlier opportunity to defect. This way of thinking can occur at every prior repetition till the first time the situation occurs, resulting in the outcome that *for all* repetitions, prisoners 'in cahoots' will feel inclined to betray the other.

Interestingly enough, this may not be the case if the game repeats infinitely. From the logic above, if the end point is never met, the prisoners have no option in which to rat the other out without incurring the punishment of the other in the subsequent repetition. Using the notation for repetitions as $t = 1, 2, \ldots$, we see that in some instances, the prisoners will rat each other out early in the game: when t is close to 1. In other instances, t may indeed be large, but the point at which the cooperation between the two falls apart will eventually make prior outcomes insignificant. For example, if the point at which the prisoners decide to rat each other out is $t' = 1$ mil, all the cooperative outcomes prior to t' are trivial when faced with the Nash equilibrium result for every t from t' to ∞. Thus, long term considerations can prove an interesting means of studying social dilemmas.

Another interesting type of social dilemma is that which is described by Levhari and

Mirman (1980) and more recently by Joosten (2007): fishery wars. In fishery wars, there is a large renewable common-pool resource from which all the fishermen obtain their catch. This is an example of a '*Commons*'-type dilemma like that described by Hardin (1968). For fisheries, the renewable common-pool resource is the oceans, seas and lakes. Short term gain here results from 'fish as much as you want' actions and can have a devastating effect on the long term possible payoffs because the resource is not given sufficient time to replenish.

1.1 Game Theory in Social Dilemmas

A unique perspective on these types of scenarios was developed by Brenner and Witt (2003) and by Joosten, Brenner and Witt (2003). By making adjustments to *repeated games*, they were able to derive a method by which the past actions of the players influence the payoffs they can receive now. This puts a rather realistic slant to the social dilemma concept. Obviously, many situations in society tend to repeat themselves (these repetitions are called *stages*), and the actions performed by the players are most likely influenced by the number of times the situation has occurred in the past, and what the outcomes had been. Joosten *et al.* (2003) coined the name 'frequency-dependent (FD)-games'. This enhances the idea that it is the *frequency* with which past stage payoffs have been achieved that can affect the current stage payoff, as opposed to situations in which the current stage payoff is uninfluenced by what has happened in the past.

Later on, a new study added further realism to the study of social dilemmas: a stochastic game model by Joosten and Meijboom (2010). Stochastic games consider situations in which the possible payoffs available, changes. Each set of payoffs, and the actions through which they are achieved, is referred to as a *state*, making a stochastic game a multi-state game. The changing from one state to another is dictated by a transition probability matrix.

Joosten and Meijboom (2010) developed their own algorithm which considered how frequency-dependent transition probabilities would affect games with frequency-independent stage payoffs. The algorithm placed particular emphasis on the effects produced in temporarily absorbing states. With no focus on absorbing states, my thesis examines communicating states only. Unlike absorbing states, communicating states are those in which the probability of leaving a particular state or of remaining in a particular state is *never* zero at any time t (for more on communicating states, see Chapter 4 in Ross (2010)). Obtaining the necessary frequency-dependent rewards to fit

their model cost Joosten and Meijboom significant processing time especially in what I refer to as their 'Type III' model (see Table 1.1); a point that will be elaborated on in the *Conclusion*, after my algorithm has been explained.

Mahohoma (2014) considered stochastic games with frequency-*in*dependent transition probabilities, yet the stage payoffs *are* frequency-dependent. To avoid confusion, it seems appropriate to summarise previous research in this area in a table: Table 1.1.

Game Type	**Type I**	**Type II**	**Type III**
States*	*1*	≥ 2	≥ 2
Play	*Repeated*	*Stochastic*	*Stochastic*
Payoff, θ	1) θ_0 2) $\theta(x)$	1) θ_0 2) $\theta(x)$	1) θ_0 2) $\theta(x)$
Trans. Prob.	n/a	p_0	$p(x)$

Table 1.1: A Table of Games. $\theta(x)$ and $p(x)$ represent the dependency on the history of play of the payoff and transition probability respectively. Literature has been rigourous enough to encompass 2+ player models. For simplicity in model-building, this thesis is restricted to 2-player games.

There[1] , using my notation one can see that Joosten *et al.* (2003) did work in Type I [$\theta(x)$] games. Mahohoma's work (2014) was in Type II [$\theta(x), p_0$] games, with Joosten and Meijboom (2010) in Type III [$\theta_0, p(x)$] games. In this Master thesis I take the next step in model development: Type III [$\theta(x), p(x)$] games.

1.2 Thesis Aim and Paper Organisation

The aim of this thesis is to *create a model that can commute multiple reward sets of a 2-player, 2-action, 2-state stochastic game with communicating states and frequency-dependent transition probability and/or frequency-dependent stage payoffs*. As a step-by-step approach is taken in the

[1]Note that this table is a work in progress. The initial understanding of what differentiated a repeated game from a stochastic game has since changed: repeated is a type of stochastic, and not separate as shown in the table. However, as the table still allows for an understanding of the work performed, it will remain with alterations given in the *Further Developments* section of this thesis.

algorithm's development, a by-product of this aim will be the development of, not just one, but a set of Matlab© algorithms that encompasses all the computations and methodology noted in Table 1.1.

With these thoughts in mind, this paper is organised as follows: the methodology is developed in Section 2. This is followed by Section 3: Results and Discussion. The final part of the main part of this thesis writeup is Section 4 which presents ideas on future work and brings all thoughts together in a conclusion. The appendix section contains the .m algorithms developed.

2. Methodology

In a single-play competitive game, there is only one instance in which the players decide their action choices (an '*action pair*' for a 2-player game like the one studied here). This single instance results in a single payoff to each player. As stated in the introduction, a repeated game contains multiple instances called *stages* where the players choose their actions. These stages follow discrete time step $t = 1, 2, \cdots, T$, with each stage having its own *stage payoff*. In this section I explain the relationships among stage payoffs, rewards and reward sets, the frequency matrix and the transition probabilities in repeated and stochastic games, as generated by the algorithm.

2.1 Necessary Conditions

In order to employ the methodology employed in Joosten *et al.* (2003), there are a few requirements that must be explained.

2.1.1 Jointly-Convergent Pure Strategies

On *pure strategies*. If at stage t, a player's action is chosen with 100% probability, it is called a *pure* action. A set of a player's pure actions is a *pure strategy*. That is, at every stage of the play, in every state of the game, the player uses one of the available actions with probability 1.

On *jointly-convergent* strategies. A matrix of frequencies corresponding to the matrix of payoffs

(see Fig. 2.1) is written as

$$\mathbf{X}^t = \begin{bmatrix} x_1^t & x_2^t \\ x_3^t & x_4^t \end{bmatrix} \tag{2.1}$$

For example, for a game with 1000 repetitions having taken place, players choices have set the game play in the upper left element 100 times, the lower left elements 3 times, the upper right element for 850 times and 47 times in the lower right element is described by:

$$X^{1000} = \begin{bmatrix} \frac{100}{1000} & \frac{850}{1000} \\ \frac{3}{1000} & \frac{47}{1000} \end{bmatrix}, \tag{2.2}$$

so that for this game,

$$X^{1000} = \begin{bmatrix} 0.100 & 0.850 \\ 0.003 & 0.047 \end{bmatrix}. \tag{2.3}$$

The condition of Eq. (2.4) becomes obvious:

$$\sum_{i=1}^{4} x_i^t = 1 \tag{2.4}$$

The t here labels all the times the payoff corresponding to that position in the payoff matrix was obtained up till time t. Additionally, the t superscript indicates that for every stage, there exists a different value for x_i^t. For example, if after only 100 iterations of the game just described,

$$X^{100} = \begin{bmatrix} 0.00 & 0.90 \\ 0.02 & 0.08 \end{bmatrix} \tag{2.5}$$

then we see that $X^{100} \neq X^{1000}$. In general X^t may not necessarily be equal to X^{t+1}. When a strategy is said to *converge*, each of these four values converges to a certain four numbers for every t in the long run (i.e., for $T \to \infty$). Joosten *et al.* (2010) formalise this from the perspective of the players' strategies.

If π represents the strategy used by player 1 (e.g. $\pi=$ action 1, action , 2 action 2, action 1, for a four stage game) and σ represents the strategy used by player 2, we can say the following: A strategy pair (π, σ) is *jointly convergent* if

$$\forall_{\epsilon>0} \forall_i \quad \limsup_{t\to\infty} \mathbf{Pr}_{\pi,\sigma}[|x_i^t - x_i| \geq \varepsilon] = 0 \tag{2.6}$$

$\Pr_{\pi,\sigma}$ represent the probability associated with the strategies of both players. '$\limsup$' is the limit superior with any $\varepsilon > 0$. Eq. (2.6) allows for the following matrix definition:

$$\begin{bmatrix} x_1^t & x_2^t \\ x_3^t & x_4^t \end{bmatrix} \xrightarrow{(T\to\infty)} \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \tag{2.7}$$

where $\sum_{i=1}^{4} x_i = 1$.

2.1.2 Long Run Average Reward

There are various ways of evaluating the scenario in which $T \to \infty$ in repeated and stochastic games. A simple one would be to add up all the stage payoffs. Here, like in Joosten *et al.* (2003), I use the *long run average reward*:

$$\vartheta^k(\pi,\sigma) = \liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \vartheta_t^k(\pi,\sigma) \tag{2.8}$$

where (π,σ) represent the strategies of players and $\vartheta_t^k(...)$ is the expected payoff at stage t to players $k = 1, 2$. '$\liminf$' is the limit inferior. We use the '$\liminf$' as the average need not exist.

2.2 Repeated Games

2.2.1 Payoff Matrix and Frequency-Dependent Rewards

The setup for a 2-player, 2-action repeated game is shown in Fig. 2.1. The payoff per stage of the

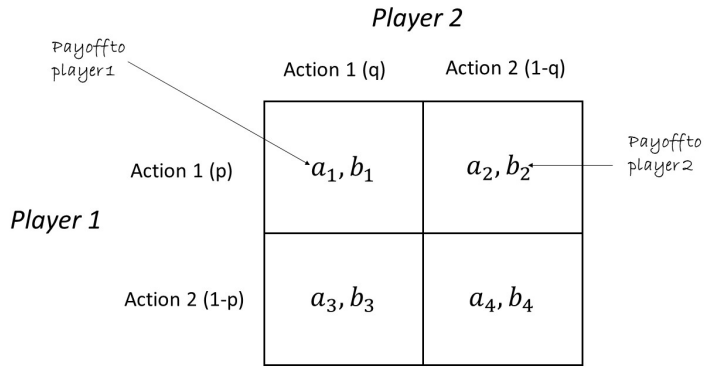


Figure 2.1: A Competitive Game

game is a choice of one of the elements of the matrix in Fig. 2.1 dictated by the actions of both

players at that stage. The *average reward* until stage t in a 2-player game with jointly converging strategies is provided by a combination of the stage payoffs and the stage frequencies of $\mathbf{X}^t$ when t is large:

$$\begin{aligned}\theta^t[pl_1, pl_2] &= x_1^t(a_1, b_1) + x_2^t(a_2, b_2) + x_3^t(a_3, b_3) + x_4^t(a_4, b_4) \\ &= \sum_{i=1}^{4} x_i^t \cdot \theta_i \end{aligned} \tag{2.9}$$

where $\theta_i = (a_i, b_i)$ and pl_1^t refers to the average payoff for Player 1. If we set limits as $t \to 0$, we get Eq. 2.8 The superscript on θ means that after t iterations, there is a *set of rewards* that the players have obtained for that game. This reward set can be used to make a *reward space* in $\mathbb{R}^2$.

In order to ensure that the code is functional, the algorithm is tested using

$$\Theta = \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix} = \begin{bmatrix} 16, 16 & 14, 28 \\ 28, 14 & 24, 24 \end{bmatrix} \tag{2.10}$$

Like the Prisoner's Dilemma, the lower right value is the Nash equilibrium. However, unlike the Prisoner's Dilemma, this value *is* Pareto superior to ('dominates') the upper left value which is now its Pareto inferior. The Pareto superior value represents 'abuse' of a hypothetical renewable common-pool resource. That is, the resource cannot sustain such a large number being removed consistently over time. On the other hand, the Pareto inferior value in the upper left of the matrix does allow for a renewability of the resource. In this manner, Eq. (2.10) models a '*Commons*' dilemma. We can consider the Pareto inferior to be the long term 'best for both' (BB) payoff pair, and the Pareto superior to be the short term BB payoff pair.

2.2.2 Frequency-Dependent (FD) Payoffs

Obtaining any of these payoffs before current time t, affect the common-pool total. This aspect is considered through a *frequency-dependent function*. FD function, for short. Section 2 of Joosten (2016) builds a methodology explaining how the FD is formulated. With the FD, the payoff is seen as a fractional value of the reward in Eq. 2.9. Mainly,

$$\text{Payoff } = FD \cdot \theta^t[pl_1, pl_2] \tag{2.11}$$

For the purposes of algorithm building, I use a simple linearly decreasing function of the form:

$$FD = 1 - \gamma_1(\gamma_2(x_2) + \gamma_3(x_3)) - \gamma_4(x_4) \tag{2.12}$$

The absence of x_1 in the formula emphasises the idea that an ecologically-friendly approach to fishing should not affect the stock; $\gamma_4 > \gamma_1 \in [0, 1]$ because both players abusing the resource simultaneously is more harmful to the fish stock than individual abuse. The restrictions on γ_2 and γ_3 are that

1. they are both non-negative, and
2. $(\gamma_2 x_2 + \gamma_3 x_3) < \gamma_1^{-1}$

$\gamma_2 \neq \gamma_3$ represent situations in which there is only one abuser who does more damage to the fish stock than the other when the other is the lone abuser.

For the Type I games produced here, let

$$FD_I = 1 - \frac{1}{4}(x_2 + 2x_3) - \frac{2x_4}{3} \tag{2.13}$$

The subscript represents the game type to which the FD function is applied.

2.3 Stochastic Games

Stochastic games contain multiple states (payoff matrices) and a transition probability matrix, that determines which of these states the play will move from stage (t) to stage $(t + 1)$. This makes a repeated game a stochastic game with a transition probability of 1 that the stage $(t + 1)$ remains in the current state.

2.3.1 Multiple States

As Table 1.1 showed, the Type II and Type III games studied here have at least two states each. Joosten and Meijboom (2010) refer to these as *High* and *Low* states. For this study, $High$ corresponds to S_1 - the situation in which renewability of the resource can occur; and Low corresponds to S_2 - the situation in which renewability of the resource is not given sufficient time to replenish itself.

To test the algorithm, these values were used:

$$\Theta_{S_1} = \begin{bmatrix} 16, 16 & 14, 28 \\ 28, 14 & 24, 24 \end{bmatrix} \qquad \Theta_{S_2} = \begin{bmatrix} 4.0, 4.0 & 3.5, 7.0 \\ 7.0, 3.5 & 6.0, 6.0 \end{bmatrix} \tag{2.14}$$

The repeated game mentioned earlier can be made by setting the probability of moving to S_2 as zero for all t.

2.3.2 Transition Probability Matrix

The transition probability matrix determines with what probability the play will move from one state to another (or the same state) for the next stage. For a 2-state system, each matrix contains elements that read as:

(*probability of moving to* S_1 *at* $t+1$ **,** *probability of moving to* S_2 *at* $t+1$)

where both values sum to 1.

The numerical example used here is:

$$p_0^{S_1} = \begin{bmatrix} 0.8, 0.2 & 0.7, 0.3 \\ 0.7, 0.3 & 0.6, 0.4 \end{bmatrix} \qquad p_0^{S_2} = \begin{bmatrix} 0.5, 0.5 & 0.40, 0.60 \\ 0.4, 0.6 & 0.15, 0.85 \end{bmatrix} \tag{2.15}$$

where the subscript (0) represents a transition probability that is independent of the frequencies (non-FD).

For ease of use, the matrices are re-written as a single vector containing only the probabilities of transitioning to S_1 at $t+1$ for the algorithm.

$$p_0 = \begin{bmatrix} 0.8 & 0.7 & 0.7 & 0.6 & 0.5 & 0.4 & 0.4 & 0.15 \end{bmatrix} \tag{2.16}$$

2.3.3 Frequency-Dependent Stage Payoffs in Stochastic Games

Multiple-states means more x values are required. For a 2-state, 2-action, 2-player system where $t \to \infty$,

$$\sum_{i=1}^{8} x_i^t = 1 \tag{2.17}$$

resulting in two matrices $\mathbf{X}_1^\infty$ and $\mathbf{X}_2^\infty$, where the subscripts refer to the state which they represent. As shown in Eq. (2.18), the frequency vector is an element of the 7-dimensional space Δ^7.

$$x^t \in \Delta^7 = \{x \in \mathbb{R}^8 | x_i \geq 0 \text{ for all } i = 1, \ldots, 8 \text{ and } \sum_{i=1}^{8} x_i = 1\} \tag{2.18}$$

For a 2-player game, this frequency point is projected onto a 2-D plane. This results in some interesting plots, as will be seen in the results.

The FD function corresponding to the example being described here should also include these additional x-terms. Again, I use a simple linear function, this time a version specific for Type II and Type III games. Let:

$$FD = 1 - \gamma_1(\gamma_2(x_2) + \gamma_3(x_3)) - \gamma_4(x_4) - \gamma_5(\gamma_6(x_6) + \gamma_7(x_7)) - \gamma_8(x_8) \tag{2.19}$$

Along with the restrictions as stated for Eq. 2.12, x_5 is not present as it does not affect the resource, and $\gamma_4 < \gamma_8$ and $\gamma_1 < \gamma_5$. This latter restriction is due to the fact that abuse in state 2 is worse than abuse in state 1 because state 2 contains less of the resource than state 1.

For the algorithm, I employ the following example, based on Eq. 2.19:

$$FD_{II,III} = 1 - \frac{1}{4}(x_2 + x_3) - \frac{x_4}{3} - \frac{1}{2}(x_6 + x_7) - \frac{2x_8}{3} \tag{2.20}$$

2.4 Computing Type II and Type III Games

To recap, there are three types of games (I, II, III). The first contains a single matrix (state), the other two contain two matrices (in my example, but can contain more) of possible payoffs for 2 players who each have two action choices that can be played at time t. There is an average reward corresponding to each pair of actions. Using an FD- function on the average reward at time t, one can obtain the payoff specific to that time, given the history of play *up to* time t.

The next step: how to determine what the history of play was? That is, what are the elements of matrix $\mathbf{X}^{\infty}$?

2.4.1 Random Sampling

I obtain values for the $\mathbf{X}^{\infty}$ matrices via the β-distribution. A β-distribution is a distribution which contains all possible values of an unknown probability. As interesting things tend to happen with the smallest probability, it may prove most interesting to use the least likely probabilities (that is, those close to 0 and 1). Additionally, focusing on the least likely values, will provide a broad range of reward set values the quickest (faster than the uniform distribution). To do this, the variables for the β-distribution are set to $\frac{1}{2}$. This increases the likelihood of choosing values close to the edges of the distribution (Fig. (2.2)). A distribution of this type is referred to as a *U-shaped Distribution*.

2.4.2 Random Sampling and the Transition Probability

A constraint to these randomly chosen x values is that they must fulfill the *flow equation*.

$$\sum_{i=1}^{4} x_i(1 - p_i) = \sum_{i=5}^{8} x_i p_i \tag{2.21}$$

where i refers to the vector's element number. If the flow equation holds, the system's states can be classified as communicating in the long run. Thus, $\sum_{i=1}^{4} x_i(1-p_i) \neq 0$ and $\sum_{i=5}^{8} x_i p_i \neq 0$, which

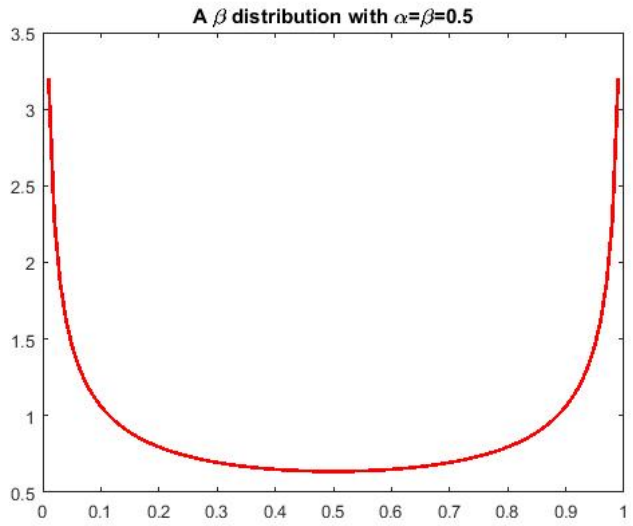


Figure 2.2: The U-shaped β-distribution is made when both α and β are set to one half.

results in either state 1 or state 2 (respectively) being an absorbing state.

In order to perform this calculation, usage of an intermediate vector y and a dummy variable Q are required to obtain the necessary degree of freedom. The values that fit the Eq. (2.21) are stored in the vector x^*. For S_1:

$$y_i = \frac{x_i}{\sum_{j=1}^{4} x_j} \qquad i = \{1, 2, 3, 4\} \tag{2.22}$$

and for S_2

$$y_i = \frac{x_i}{\sum_{j=5}^{8} x_j} \qquad i = \{5, 6, 7, 8\} \tag{2.23}$$

y is related to x^* via:

$$x^* = \begin{bmatrix} Qy_1 & Qy_2 & Qy_3 & Qy_4 & (1-Q)y_5 & (1-Q)y_6 & (1-Q)y_7 & (1-Q)y_8 \end{bmatrix} \tag{2.24}$$

Using Eq.s (2.23) and (2.22), Eq. (2.21) in terms of y gives

$$Q \sum_{i=1}^{4} y_i(1-p_i) = (1-Q) \sum_{i=5}^{8} y_i p_i \tag{2.25a}$$

$$Q = \frac{\sum_{i=5}^{8} y_i p_i}{\sum_{i=1}^{4} y_i(1-p_i) + \sum_{i=5}^{8} y_i p_i} \tag{2.25b}$$

Once Q is determined (and by extension, $1-Q$), it is a simple matter of utilizing Eq. (2.24) once again, to solve for x^*.

This way, one can see that I move from the randomly (β-distribution) chosen values of $\mathbf{X}^{\infty}$ to an intermediate matrix y, to solving for Q, and finally obtaining the corrected $\mathbf{X}^{\infty}$ matrix that transitions from S_1 to S_2 and back: $x^* = x(y, Q)$.

In the case (of 2+ states), Eq. (2.25b) generalises to a system of linear equations.

2.4.3 Frequency-dependent Transition Probability

The addition of frequency-dependence on the transition probability matrix requires a bit more refinement to the steps previously presented. Now there is an additional complexity that,

$$p(x) = p_0 - \Big[x \cdot A\Big] \tag{2.26}$$

where the 8x8 matrix A should contain non-negative values and $\Big[x \cdot A\Big] < p_0$.

The example of A used is:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .35 & .30 & .30 & .25 & .20 & .15 & .15 & .05 \\ .35 & .30 & .30 & .25 & .20 & .15 & .15 & .05 \\ .70 & .60 & .60 & .50 & .40 & .30 & .30 & .10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .35 & .30 & .30 & .25 & .20 & .15 & .15 & .05 \\ .35 & .30 & .30 & .25 & .20 & .15 & .15 & .05 \\ .70 & .60 & .60 & .50 & .40 & .30 & .30 & .10 \end{bmatrix} \tag{2.27}$$

Note that the 1^{st} and 5^{th} rows in Eq. (2.27) correspond to eco-friendly playing strategies: x_1^t and x_5^t. As an example of how Eq. 2.26 works, if

$$x = \Big[0.175 \quad 0.025 \quad 0.025 \quad 0.175 \quad 0.450.025 \quad 0.1 \quad 0.025\Big], \tag{2.28}$$

then, for the first entry of the 1x8 array $x^*.A$:

$$\begin{bmatrix} 0.175 & 0.025 & 0.025 & 0.175 & 0.450.025 & 0.1 & 0.025 \end{bmatrix} * \begin{bmatrix} 0 & \cdots \\ 0.35 & \cdots \\ 0.35 & \cdots \\ 0.70 & \cdots \\ 0 & \cdots \\ .35 & \cdots \\ .35 & \cdots \\ .10 & \cdots \end{bmatrix} = [0.2012 \cdots] \quad (2.29)$$

Given Eq. 2.16, Eq. 2.26 becomes

$$p(x) = \begin{bmatrix} 0.59875 & 0.5275 & 0.5275 & 0.45625 & 0.385 & 0.31375 & 0.31375 & 0.0925 \end{bmatrix} \quad (2.30)$$

However, Eq. 2.28 corresponds to p_0, so that $p(x)$ may well correspond to another x. Thus, a re-calculation of x is required.

The entire process flows as follows: I first choose a random set of numbers for the $\mathbf{X}_1^t$ and $\mathbf{X}_2^t$ matrices keeping Eq. (2.17) in mind. I then use these matrices to obtain both the intermediate y vector as well as $p(x)$. y and $p(x)$ are used to solve for Q (or vector of Q in the general case). This leads to x^*.

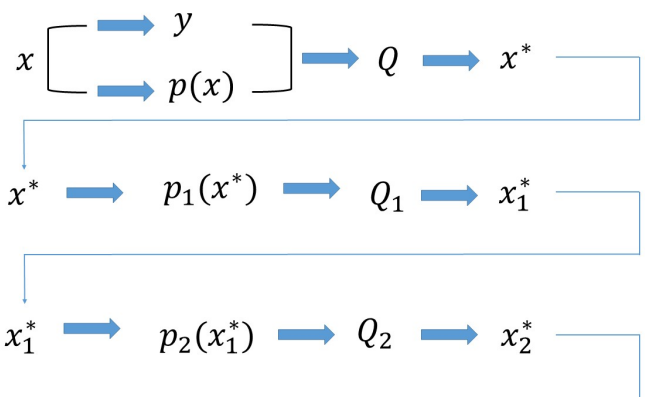


Figure 2.3: A flowchart showing the beginning of the iterative process involved in solving for x^* in Type 3 games. y is kept constant throughout.

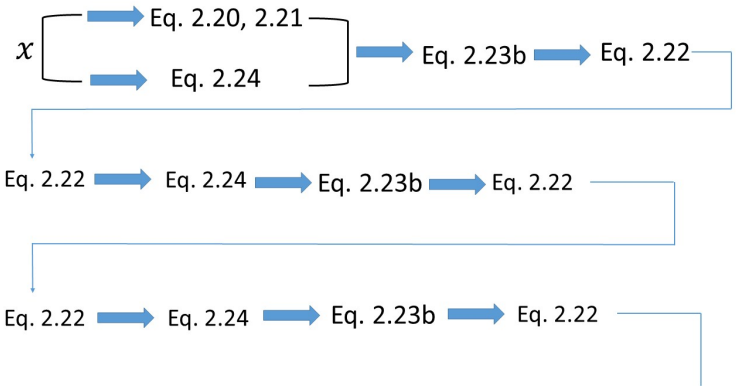


Figure 2.4: A flowchart showing the equations used to begin the iterative process. x comes from the random sampling described in Section 2.4.1.

This process is repeated again and again (as seen in Fig. 2.3, with corresponding equations in Fig. 2.4) resulting in a vector of values: $Q = [Q_1, Q_2, \ldots, Q_{n-1}, Q_n]$ where n is the iteration number of the 'Q'-loop. These values exhibit linear convergence and so the loop continues until there is a trivial difference between Q_{n+1} and Q_n, calculated via basic subtraction.

To make the algorithm faster, I increase the speed of the convergence of this sequence through the use of the 'Aitken's Δ^2 method' (Equation 2.14 in Burden and Faires (2010)) which here takes the form,

$$Q^{check} = Q_n - \frac{(Q_{n+1} - Q_n)^2}{Q_{n+2} - 2Q_{n+1} + Q_n} \tag{2.31}$$

The condition I use to stop the iterations is

$$(Q^{check}_{m+1} - Q^{check}_{m}) < 1.0 \times 10^{-8} \tag{2.32}$$

Note that from Eq. (2.31), $m \neq n$ since n must reach $n = 3$ before $Q^{check}_{m=1}$ can be found.

Thus, instead of subtracting every subsequent Q, I instead use Eq. (2.31) which determines significant differences much sooner and for smaller levels of precision.

Standard Subtraction to the 10^{-6}	Aitken's Method to the 10^{-8}
$\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \\ Q_8 \end{bmatrix}$	$\begin{bmatrix} \\ \\ Q_1^{check} \\ Q_2^{check} \\ Q_3^{check} \\ \\ \\ \\ \end{bmatrix}$

Table 2.1: Method for Determining the 'true' Q. In this example (based on initial calculation done by hand), subsequent differences in Q to precision 10^{-8} *are zero by* $m = 3$ *using the Aitken's method, but not so till* $n = 8$ *using standard subtraction to precision* 10^{-6}.

3. Results & Discussion

3.1 Type I Games

In addition to providing the foundation for the development of the algorithm, Type I games provide an interesting insight into the effects of the frequency matrix on dynamic competitive games.

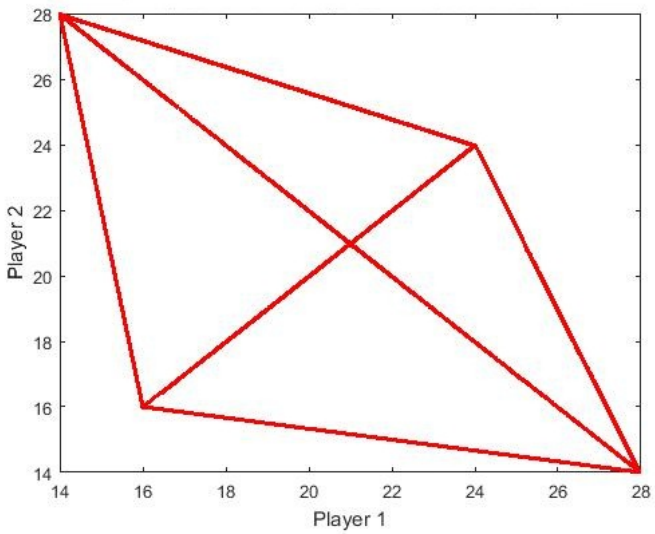


Figure 3.1: A line plot for a Type I Game with non-FD stage payoffs.

The line plot (as seen in Fig. 3.1 for example) was created to give as clear an idea as possible as to the shape of the reward set in $\mathbb{R}^2$. It was made by setting 2 out of the 4 elements of x values in Eq. (2.9) to zero. For example, if $x_1 = x_2 = 0$, and random values of x_3 and x_4 are made using a

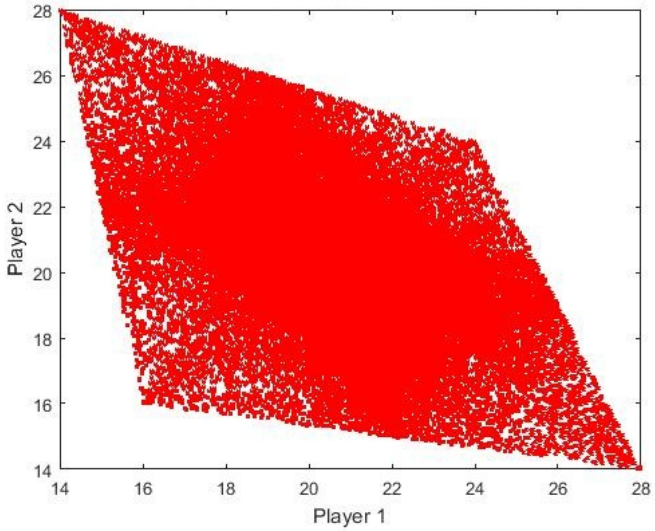


Figure 3.2: A scatter plot for a Type I Game with non-FD stage payoffs.

$\beta-$distribution, I obtain the line connecting $(28, 14)$ to $(24, 24)$. That is,

$$\theta^t[pl_1, pl_2] = 0 \times (16, 16) + 0 \times (14, 28) + x_3^t \times (28, 14) + x_4^t \times (24, 24) \tag{3.1}$$

for all $t = \{1 \cdots, T\}$ In cases of the line plot $T_L = 20,000$. For the scatter plots, $T_S = 50,000$. $T_L \neq T_S$ for the simple reason that, with so many zero values created when generating a line plot, the extra $30,000$ add nothing new to these images.

The scatter plot version of Fig. 3.1 is seen in Fig. 3.2. Through it, one is able to visualise that for random samples of $\mathbf{X}^\infty$ when t is large, there are higher concentrations of rewards closer to the midpoint of the lines provided in the line plot.

Type I FD Game

The plot in Fig. 3.3 is particularly insightful. By manipulating FD_1, we can see how the reward set shifts as the rewards it represents becomes increasingly influenced by FD_I. I did this by inserting a dummy variable $\alpha \in [0, 1]$ into Eq. (2.12) such that:

$$FD_I = 1 - \frac{\alpha}{4}(x_2 + 2x_3) - \frac{2\alpha}{3}x_4 \tag{3.2}$$

By increasing the α through 5 runs of the algorithm (i.e. 5 different games), I create 5 different sets of rewards, each with its own visual characteristics. As the only x_i that remains unaffected is x_1, we see that (16,16) remains fixed for all 5 games.

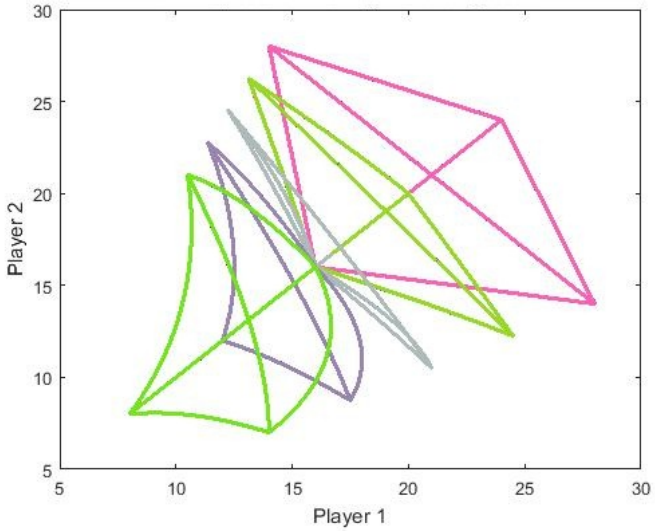


Figure 3.3: Line plots of 5 Type I games indicating the change in reward sets from one game to the next. As the effect of the FD_I function on the reward sets increase from game to game, the reward space folds over so that the maximum upper right point becomes the lowest, lower left point. This is done by increasing the value of α in Eq. (3.2). The extremes $\alpha = 0$ and $\alpha = 1$ are shown in Fig. 3.4.

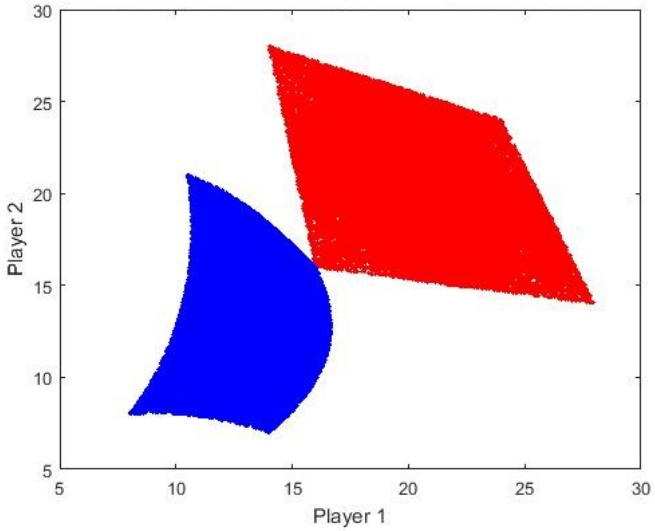


Figure 3.4: The scatter plots compare the reward set of a Type I non-FD game (north-east in red) and the reward set of a Type I FD game (south-west in blue)

In terms of the Pareto points, clearly the larger values (north east in the plot) become steadily less and less obtainable as a result of player resource-abuse. These larger values are near the short term (Pareto superior) payoff point. As the full weight of FD_I comes into play, the reward

set itself *folds over* on the long term (Pareto inferior) point $(16, 16)$ so that it becomes the maximum point in the final run of the algorithm where $\alpha = 1$. In Fig. 3.3 we see this as $(24, 24)$ reduces, becoming $(4, 4)$.

Thus within this setup, one can say that with constant abuse of a common-pool resource over time, the Pareto superior point reduces in value till it becomes the Pareto inferior point. This new Pareto inferior point is still larger than the new Pareto superior point which has become even smaller.

Note that Fig. 3.4 is in keeping with a linear-reducing shift of the rewards as seen in Fig. 1 of Joosten *et al.* (2003).

3.2 Type II Games

Recall that Type II games are those in which there are two states that possess frequency-dependent payoffs and frequency-*in*dependent transition probabilities: Type II parameters are $[\theta(x), p_0]$.

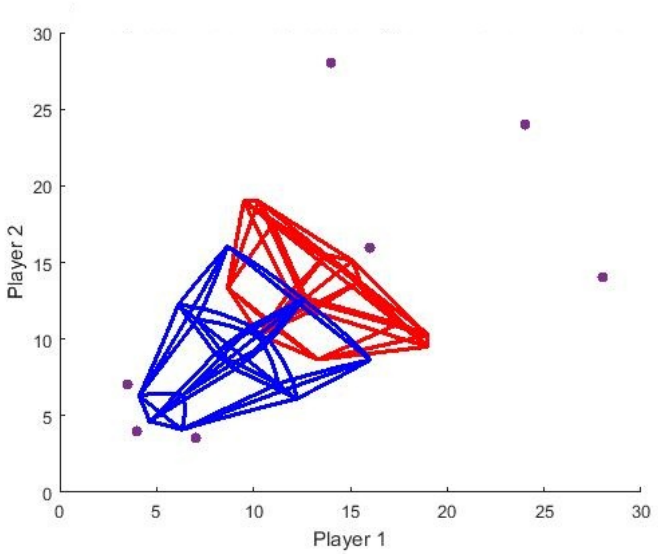


Figure 3.5: The (red) line plot to the north-east is that of the Type II non-FD game. The blue line plot to the south-west is the Type II FD game. The dots indicate the location of the stage payoffs of S_1 and S_2.

Manipulating the number of zero elements as described earlier allows for the visualization of different aspects of the reward set. In Fig. 3.5, two elements in $\mathbf{X}_i^{\infty}$ and 1 element in $\mathbf{X}_j^{\infty}$ ($i = 1, 2\ , j = 1, 2\ , i \neq j$) is used consecutively. For Fig. 3.6 and Fig. 3.7, two elements in both

$\mathbf{X}_1^\infty$ and $\mathbf{X}_2^\infty$ are set to zero.

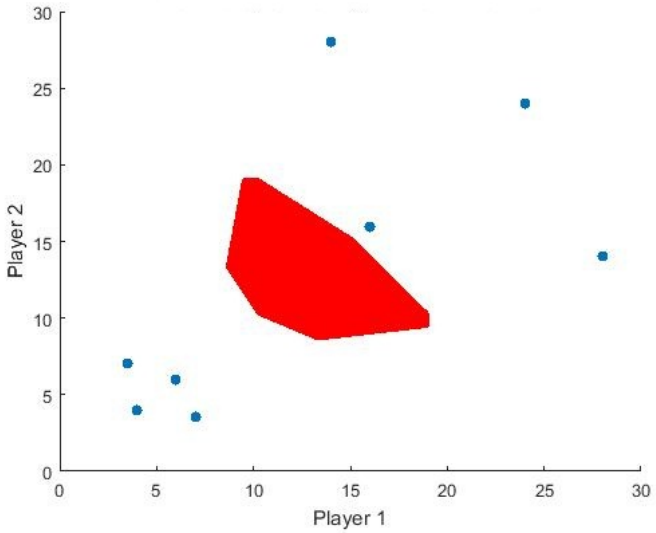


Figure 3.6: Here is a filled 2D version of the reward set when the stage payoffs are non-FD. Two x values in each state's frequency matrix are set to zero here. The dots indicate the location of the stage payoffs of S_1 and S_2.

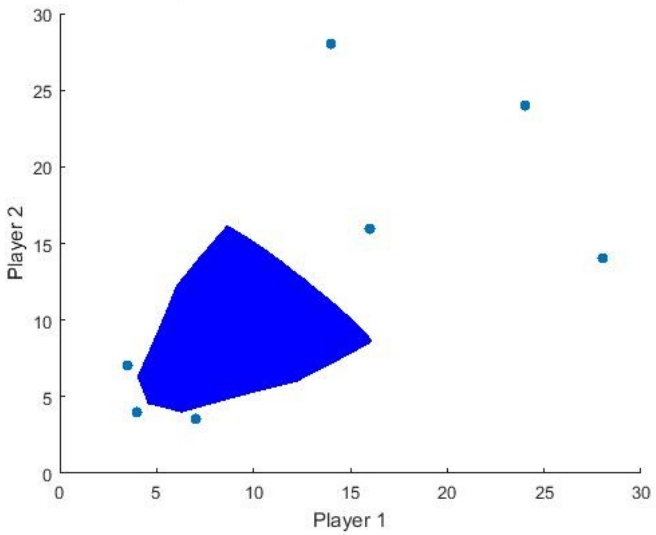
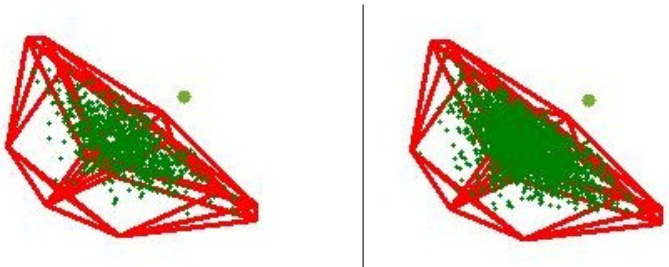


Figure 3.7: Here is an entire reward set when the stage payoffs are FD. Two x values in each state's frequency matrix are set to zero here. The dots indicate the location of the stage payoffs of S_1 and S_2.

In this example, unlike its Type 1 counterpart, the reward set for Type II games do not fold over on a point, but instead seems to shifts downward. During this shift, the set becomes elongated



(a) 1000 iterations *(b) 2500 iterations*

Figure 3.8: Type II Game. Cropped images of scatter plots of reward sets for (left) 1000 iterations and (right) 2500 iterations atop the line plots of the Type II non-FD reward space. The area of highest concentration is close to the north-east face of the reward set.

and retains some of its lower-valued non-FD reward pairs.

By changing the number of iterations in a game, as in Figures 3.8 and 3.9, the areas of highest concentration (highest probability 'best for both' points) can be determined. For the non-FD version, it lies around (14, 14). For the FD version it lies lower, around (10, 10).

It should also be noted that this area of highest concentration forms around a central region where the majority of the outlines intersect. As this inner region stretches during the transition from non-FD to FD, so too does the highest concentration of 'best for both' points.

In both FD and non-FD cases, both players seem to have equally likely chances of gaining a better payoff than their opponent.

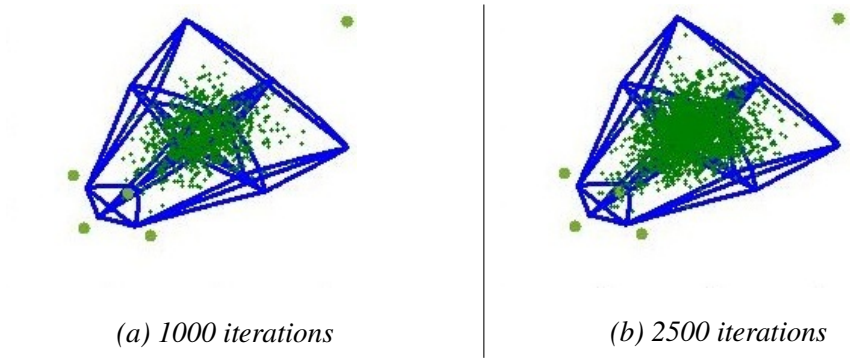
To summarise, the effect of $FD_{II,III}$ in Type II games is such that:

1. *it reduces the probability of higher (closer to S_1) rewards*,
2. *it also reduces the amount by which a player can best their opponent,*
3. *it increases the volatility of the rewards* [2].

3.3 Type III Games

Type III games are those in which there are two states that possess both frequency-dependent payoffs and transition probabilities: Type II parameters are $[\theta(x), p(x)]$. The setup for obtaining the

[2] Volatility is defined as 'the measure of the uncertainty about the rewards provided by the resource' (Section 14.4 in Hull (2012)).



(a) 1000 iterations *(b) 2500 iterations*

Figure 3.9: Type II Game. Cropped images of scatter plot of reward sets for (left) 1000 iterations and (right) 2500 iterations atop the line plots of the Type II FD reward space. The area of highest concentration is around the centre of the reward set.

variations in plots for Type III games is the same as that described for Type II games.

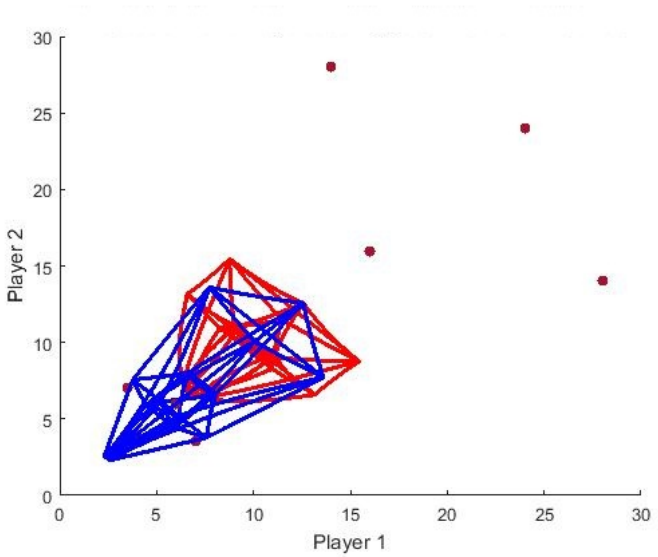


Figure 3.10: The broader line plot to the upper right (red) is the reward set for the non-FD game. The lower left (and narrower) line plot (blue) is the reward set for the FD game.

In this example, for Type 3 games, the frequency-dependency transition probability function $p(x)$ has resulted in a situation vaguely similar to the Type I model where the non-FD and FD reward sets noticeably share a 'best for both' point. Unlike with the Type I, however, and similar to the Type II, the set does not fold over from this point but elongates the non-FD *from it* to form the FD reward set. A visual determination puts this point around (14, 14). Thus the lowest 'best for both' point in S_1 becomes the highest 'best for both' point for the Type III game; with its lowest

'best for both' point going well below the lowest 'best for both' point of S_2.

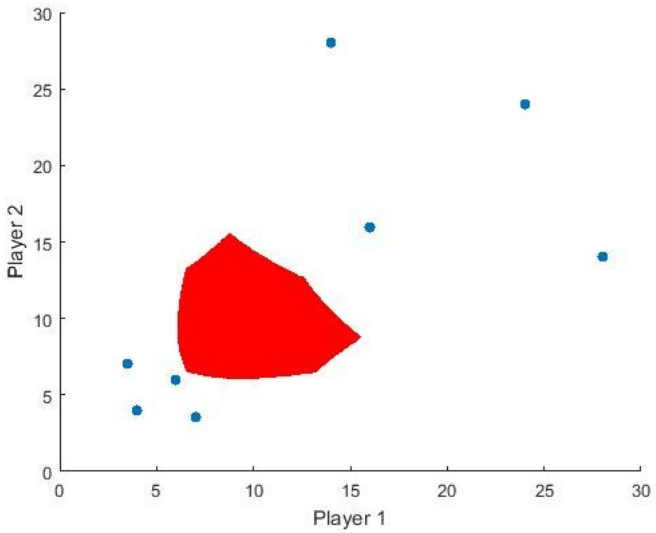
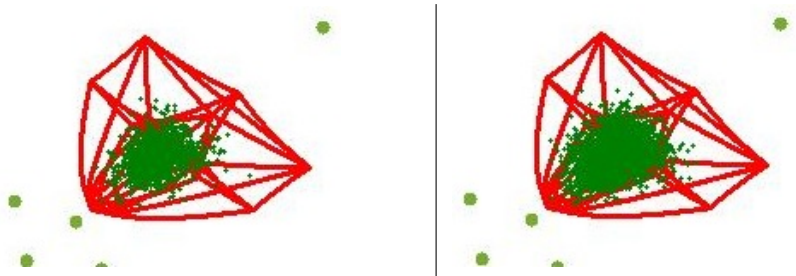


Figure 3.11: Here is an entire reward set when the stage payoffs are non-FD. Two x values in each state's frequency matrix are set to zero here. The dots indicate the location of the stage payoffs of S_1 and S_2.

On examining the regions of highest concentration, one can see from Fig 3.13 and Fig. 3.14 that the FD version has increased chances of lower rewards; much more so that its Type II counterpart. This seems to correspond with the elongation of the region of highest concentration south-westward. This means that the central region of the reward set in a Type III game is significantly affected by a frequency-dependence on the transition probability.



(a) 1000 iterations *(b) 2500 iterations*

Figure 3.13: Type III Game. Cropped images of scatter plots of reward sets for (left) 1000 iterations and (right) 2500 iterations atop the line plots of the Type III non-FD reward set. The area of highest concentration is around the centre the reward set, extending rather broadly but with some emphasis south west.

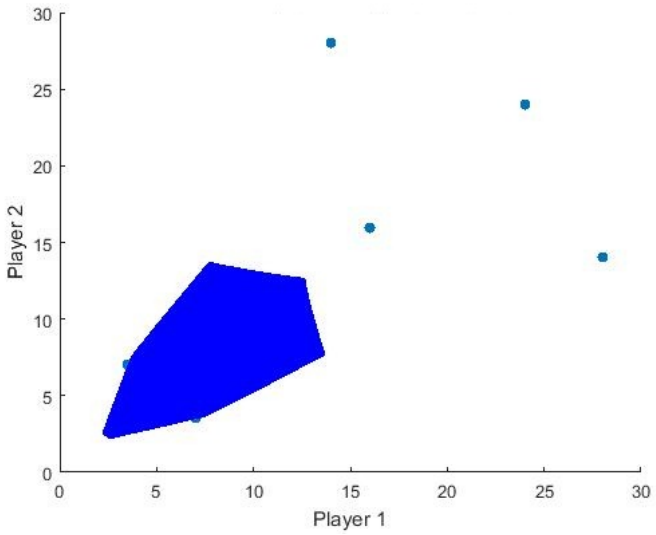
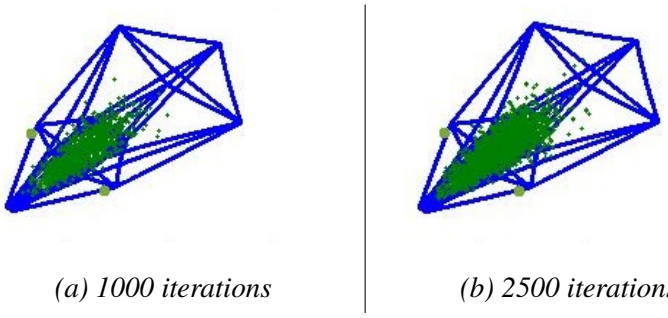


Figure 3.12: Here is an entire reward set when the stage payoffs are FD. Two x values in each state's frequency matrix are set to zero here. The dots indicate the location of the stage payoffs of S_1 and S_2.



(a) 1000 iterations *(b) 2500 iterations*

Figure 3.14: Type III Game. Cropped images of scatter plots of reward sets for (left) 1000 iterations and (right) 2500 iterations atop the line plots of the Type 3 FD reward set. The area of highest concentration is in the bottom third of the set and extends both to the south-west and to the north east.

To summarise, in the example of Type III games modeled here,

1. *FD and $p(x)$ reduces the amount by which a player can best their opponent,*
2. *$p(x)$ further increases the volatility of the rewards compared to (FD, p_0) as seen in the Type II games,* but
3. *possesses the same 'best for both' maximum for both the FD and non-FD stage reward versions.*

4. Most importantly, *the combination of $FD_{II,III}$ and $p(x)$ significantly reduces the probability of higher (closer to S_1) rewards*,

4. Conclusion

I have created a set of algorithms that allows for the representation of large sets of jointly-convergent pure strategy rewards for a large variety of stochastic games (with a repeated game being considered a special type of stochastic game). The .m files are provided in the Appendix of this document. Using Eq. (2.9) I calculate reward pairs for multiple stages. It is possible to plot these reward pairs in $\mathbb{R}^2$ to visualise a game's reward set.

With the use of an FD function, it is possible to differentiate between stage rewards that are affected by the frequency with which past actions have occurred, and those that are not so affected. It is also possible to differentiate between games in which the probability of moving from one state to another is dependent on the frequency of past actions and those that are not, through the use of a frequency-dependent transition probability function.

The model developed here is geared toward 2-player, 2-action, 2-state stochastic games. Aside from these specifications, it is actually quite broad, able to handle any type of FD function and possibly non-linear $p(x)$ functions as well. The use of the FD is entirely optional and the user can not only manipulate whether or not the stage rewards (or transition probabilities) are frequency-dependent, but can also consider the intermediate levels where these rewards (probabilities) vary in the *degree* to which they are affected.

The example used to ensure the code's functionality was the fishery war which is a '*Commons*'-type social dilemma. Here, the two players are encouraged toward actions that will harm them in the long term, as they gain in the short term. Results for Type 1 games show that the region

of highest rewards lie in the centre of the reward set's diagrammatic form. When the FD function is reduced linearly, the reward set folds over so that the Pareto inferior 'best of both' point of the initial reward space becomes the new Pareto superior 'best of both' point in the final reward set.

Results for Type II and Type III games were similar in most respects. FD stage payoffs shifted the reward set to lower values, and reduced the difference in rewards in stages where a player obtained a reward higher than the opposing player. As frequency dependency increases the 'length' of the reward set, there is a corresponding increase in the volatility of the rewards that can be obtained.

The one major area of difference between the Type II and Type III games was the re-positioning of the region of highest concentration that sat lower in the reward set's diagrammatic interpretation of the Type III model than that of the Type II, regardless of $FD_{II,III}$. Thus we can conclude that an FD transition probability is a *significant factor* in the probability of obtaining higher rewards.

Also, the Type III (FD) game shared a short term (Pareto superior) point with that of the Type III (non-FD) game.

4.1 Further Developments

Together with my supervisor, Reinoud Joosten, we are in the process of publishing the findings made in the Type III FD games, with new ideas still developing.

One major consideration throughout this project has been the classification of the different types of games. Recent discussions have led to the idea that, if repeated games are a type of stochastic game, then Table 1.1 contains an error in the 'Play' row where 'repeated' was considered as a separate class from 'stochastic'. This led to the idea of Action-Independent Transition probability matrices (A.I.T.), where the transition probability matrices in Eq. (2.15) are instead of a general form,

$$p_0^{S_1} = \begin{bmatrix} \omega, (1-\omega) & \omega, (1-\omega) \\ \omega, (1-\omega) & \omega, (1-\omega) \end{bmatrix} \qquad p_0^{S_2} = \begin{bmatrix} \rho, (1-\rho) & \rho, (1-\rho) \\ \rho, (1-\rho) & \rho, (1-\rho) \end{bmatrix} \tag{4.1}$$

where $0 \leq \omega, \rho \leq 1$. This way, no matter the action pair, the transition probabilities per state remain unchanged.

Calculation of Q here is trivial. We only need solve for Q in an equation similar to Eq.

(2.25b):

$$Q(1-\omega) = \rho(1-Q) \tag{4.2a}$$

$$Q = \frac{\rho}{1-\omega+\rho} \tag{4.2b}$$

This need only be done once for the entire game as Q here is constant.

So we see that with the concept of AIT, unlike what was stated in the Methodology: a Type I game *can* exist with multiple reachable states.

4.2 Future Work

It is important to recognise that the results in the example may not necessarily hold for all types of social dilemma. What I *can* say is that because one can model any type of social dilemma using this code, the following additions may be of academic interest:

1. Compiling all .m files into a single code with input options for the user.
2. A three-player model will allow interesting visualizations of the rewards.
3. Any of the following:
   (a) changing the type of social dilemma, Komorita and Parks (1996) give an in depth look into a few of the more well-known social dilemmas),
   (b) changing the quality of the FD functions (FD_1 and $FD_{2,3}$) or transition probability $p(x)$ function, to a non-linear (though still continuous) function, or to more involved continuous linear functions like Henry Hamburger's 'Give Some' games (see p. 12 in Komorita and Parks (1996)),
   (c) alter the frequency matrices to reflect specific strategies.
4. Creating a three-state system of S_1, S_2 and S_3. (with linear algebra, this should make moving to multi-state modeling much easier).
5. Parallel computing iterations $t = 1, ..., T$ can increase speed.
6. Consider the effects of temporarily absorbing states.

4.3 Final Thoughts

It would be remiss of me if I do not emphasise two salient points. Firstly, the code demonstrates the non-trivial effects of achieving short term gains at the cost of those in the long term. The

visualization of rewards, allowed through the algorithm I have created, presents a clear picture of the effects of abusing common-pool systems. I leave it as a project for the interested reader to add his or her functions to give realistic considerations of scenarios in which long term goals lead from low states to high ones: a much more positive spin than the scenario presented in this thesis.

Secondly, in the introduction I mentioned the efficiency of the Joosten and Meijboom (2010) Type III $[\theta_0, p(x)]$ game. My focus on communicating states (states that will never have transition probabilities equal to 0), has allowed me to develop a much more efficient algorithm to model social dilemmas than that of Joosten and Meijboom. As shown in the *Future Work* section, it is possible to adapt the code to consider absorbing states. However, whether or not the increase in processing time (as the algorithm searches for appropriate x values to fit a unique *flow equation*) will be proportional to the increase experienced by Joosten and Meijboom remains to be seen, but seems unlikely.

Overall, it is my belief that this code can form the basis for modeling social dilemmas that are either simple or extremely complex and can also consider all variations in between.

References

Brenner, Thomas and Ulrich Witt. 'Melioration learning in games with constant and frequency-dependent payoffs'. In: *Journal of Economic Behavior & Organization* 50 (2003), pp. 429–448. URL: http://dx.doi.org/10.1016/S0167-2681(02)00034-3.

Burden, Richard L. and J. Douglas Faires. *Numerical analysis*. 9th ed. Brooks/Cole, 2010.

Hardin, Garrett. 'The tragedy of the commons'. In: *Science Magazine* 162.3859 (Dec. 1968), pp. 1243–1248. URL: http://science.sciencemag.org/content/162/3859/1243.full.

Hull, John C. *Options, futures, and other derivatives*. 8th ed. Brooks/Cole, 2012.

Joosten, Reinoud. 'Small fish wars: a new class of dynamic fishery-management games'. In: *The IUP Journal of Managerial Economics* V.4 (Nov. 2007), pp. 17–30.

— 'Strong and weak rarity value: resource games with complex price–scarcity relationships'. In: *Dynamic Games and Applications* 6.1 (Mar. 2016), pp. 97–111. URL: http://link.springer.com/article/10.1007/s13235-015-0136-4.

Joosten, Reinoud, Thomas Brenner and Ulrich Witt. 'Games with frequency-dependent stage payoffs'. In: *International Journal of Game Theory* 31.4 (Jan. 2003), pp. 609–620. URL: https://doi.org/10.1007/s001820300143.

Joosten, Reinoud and Robin Meijboom. 'Stochastic games with endogenous transitions'. In: *Papers on Economics & Evolution* 4.1024 (Nov. 2010), pp. 1–29. URL: https://papers.econ.mpg.de/evo/discussionpapers/2010-24.pdf.

Komorita, Samuel S. and Craig D. Parks. *Social dilemmas*. Social Psychology Series. Westview Press, 1996.

Levhari, David and Leonard J. Mirman. 'The great fish war: an example using a dynamic cournot-nash solution'. In: *The Bell Journal of Economics* 11.1 (Spring 1980), pp. 322–334. URL: http://www.jstor.org/stable/3003416.

Mahohoma, W. 'Stochastic games with frequency-dependent stage payoffs'. diploma thesis. Maastricht University, Aug. 2014. URL: https://dke.maastrichtuniversity.nl/gm.schoenmakers/wp-content/uploads/2015/10/Master-Thesis-Mahohoma-SQ.pdf.

Peters, Hans. *Game theory. A multi-leveled approach*. Springer, 2008.

Ross, Sheldon M. *Introduction to probability models*. 10th ed. Elsevier, 2010.

Van Lange, Paul A. M., Jeff Joireman, Craig D. Parks and Eric Van Dijk. 'The psychology of social dilemmas: a review'. In: *Organizational Behavior and Human Decision Processes* 120 (Nov. 2013), pp. 125–141. URL: http://dx.doi.org/10.1016/j.obhdp.2012.11.003.

Appendices

A Algorithm 1: Type 1 Game

```
%Type 1 Game
%(0<=alpha<=1).
clear
%---[1] The payoff vectors
A1 = [16 14 28 24]';
B1 = [16 28 14 24]';
%---[2] stage level and storage
T=100000;
Payoff1 = zeros(T,1);
Payoff2 = zeros(T,1);
x = zeros(4,1);
r = zeros(4,1);
prompt = 'What is alpha?  ';
alpha = input(prompt);
for v=1:T
%---[3] Frequency vector x (scatter)
 for i = 1:4
  r(i) =betarnd(0.5,0.5);
 end
 Norm_val = sum(r);
 for i = 1:4
     x(i) = r(i)/Norm_val;
 end
%%%****
%---[3] Frequency vector x (outline)
 %for i= 1:2;
    % r(i) = betarnd(0.5,0.5);
 %end
 %Norm_val = sum(r);
 %for i = 1:2
    % x_a(i) = r(i)/Norm_val;
 %end
 %x_b = [x_a(1) x_a(2) 0 0]';
 %x = x_b(randperm(length(x_b))); %randomises x_b
    %%****
%---[4] Linear payoff function
```

```
 FD = 1-alpha*0.25*(x(2) + 2*x(3))-alpha*(2/3)*x(4);
%---[5] Stage payoff vectors
 V_p1 = x.*A1;
      V_p2 = x.*B1;
 Payoff1(v) =FD*sum(V_p1);
 Payoff2(v) =FD*sum(V_p2);
end
%---[6] plot
figure(1)
plot(Payoff1,Payoff2, '*r', 'MarkerSize', 1)
xlabel('Player 1')
axis([5 30 5 30])
ylabel('Player 2')
title(['Reward Space. T= ' num2str(T) ' iterations.'])
```

B Algorithm 2: Type 2 Game

```
%2-state stochastic game with p0. const payoff (alpha=0), FD-payoff (alpha=1).
clear
%---[1] The payoff vectors and p0
A1 = [16 14 28 24 4 3.5 7 6]'; %vector
B1 = [16 28 14 24 4 7 3.5 6]'; %vector
p = [0.8 0.7 0.7 0.6 0.5 0.4 0.4 0.15]'; %vector
    %p is the probability of moving to S1

%---[2] stage level and storage
   T = 50000; %1000000;
x = zeros(8,1);
x_a = zeros(8,1);
xstar = zeros(8,1);
r = zeros(8,1);
y = zeros(8,1);
yp = zeros(4,1);
yp_not = zeros(4,1);
v_p1 = zeros(8,1);
```

```
19 v_p2 = zeros(8,1);
20 payoff_p1 = zeros(T,1);
21 payoff_p2 = zeros(T,1);
22
23 prompt = 'What is alpha?  ';
24 alpha = input(prompt);
25
26 for t = 1:T
27 %---[3] Frequency vector x (scatter)
28       for i = 1:8
29        r(i) = betarnd(0.5,0.5);
30   end
31
32   for i = 1:8
33      x(i) = r(i)/sum(r);
34  end
35     %%****
36 %%---[3] Frequency vector x (outline)
37 %for i= 1:3 %
38 %     r(i) = betarnd(0.5,0.5);
39 %end
40 %
41 %Norm_val = sum(r);
42 %
43 %for i = 1:3 %
44 %     x_a(i) = r(i)/Norm_val;
45 %end
46 %
47 %x_b1 = [x_a(1) x_a(3) 0 0]';%
48 %x_b2 = [x_a(2) 0 0 0]';%
49 %
50 %x_c1 = x_b1(randperm(length(x_b1))); %%randomises positions of x_b
51 %x_c2 = x_b2(randperm(length(x_b2)));
52 %
53 %for i =1:8
54 %     if i< 5
55 %         x(i) = x_c1(i);
```

```
%     else
%         x(i) = x_c2(i-4);
%     end
%end
    %%****

%---[4] Intermediate y vector and Q
for i =1:4
    y(i) = x(i)/(sum(x(1:4)));
end
for i =5:8
    y(i) = x(i)/(sum(x(5:8)));
end
for i = 1:4
    yp_not(i) = y(i)*(1-p(i));
    yp(i) = y(i+4)*p(i+4);
end
Q = sum(yp)/( sum(yp)+ sum(yp_not) );
Q_not = 1-Q;

%---[5] Solving for x*
for i = 1:4
    xstar(i) = Q*y(i);
end
for i = 5:8
    xstar(i) = Q_not*y(i);
end

%---[6] FD equation
FD = 1- alpha*0.25*(xstar(2) + xstar(3))...
    -0.5*alpha*( xstar(6) + xstar(7) ) - (1/3)*alpha*xstar(4) -(2/3)*alpha*
        xstar(8);
for i= 1:8
    v_p1(i) = xstar(i)*A1(i);
    v_p2(i) = xstar(i)*B1(i);
end

```

```
92 %---[7] Stage payoff vector
93 payoff_p1(t) = FD*sum(v_p1);
94 payoff_p2(t) = FD*sum(v_p2);
95 end
96
97 %---[8] plot
98 figure(1)
99 scatter(A1,B1, 'filled')
100 hold on
101 plot(payoff_p1,payoff_p2,'o', 'MarkerSize',1,...
102     'MarkerFaceColor','r','MarkerEdgeColor', 'r')
103 xlabel('Player 1')
104 ylabel('Player 2')
105 title(['Type 2 Game Reward Space. T = ' num2str(T) ' iterations'])
106 hold off
```

C Algorithm 3: Type 3 Game

```
1  clear
2  %
3  A1 = [16 14 28 24 4 3.5 7 6]';
4  B1 = [16 28 14 24 4 7 3.5 6]';
5  p = [0.8 0.7 0.7 0.6 0.5 0.4 0.4 0.15];
6  x = zeros(1,8);
7  r = zeros(8,1);
8  y = zeros(8,1);
9  xstar = zeros(1,8);
10 x_a = zeros(8,1);
11 yp = zeros(4,1);
12 yp_not = zeros(4,1);
13 v_p1 = zeros(1,8);
14 v_p2 = zeros(1,8);
15         T = 500000;
16 payoff_p1 = zeros(T,1);
17 payoff_p2 = zeros(T,1);
```

```
18
19 matrixA = [0.00  0.0  0.0  0.00  0.0  0.00  0.00  0.00;
20            0.35  0.3  0.3  0.25  0.2  0.15  0.15  0.05;
21            0.35  0.3  0.3  0.25  0.2  0.15  0.15  0.05;
22            0.70  0.6  0.6  0.50  0.4  0.30  0.30  0.10;
23            0.00  0.0  0.0  0.00  0.0  0.00  0.00  0.00;
24            0.35  0.3  0.3  0.25  0.2  0.15  0.15  0.05;
25            0.35  0.3  0.3  0.25  0.2  0.15  0.15  0.05;
26            0.70  0.6  0.6  0.50  0.4  0.30  0.30  0.10];
27
28 prompt = 'What is alpha?  ';
29 alpha = input(prompt);
30
31 for t = 1:T
32
33     Q_checker
34
35 FD = 1- alpha*0.25*(xstar(2) + xstar(3)) -0.5*alpha*( xstar(6) + xstar(7) )...
36     - (1/3)*alpha*xstar(4) -(2/3)*alpha*xstar(8);
37 for i= 1:8
38     v_p1(i) = xstar(i)*A1(i);
39     v_p2(i) = xstar(i)*B1(i);
40 end
41 payoff_p1(t) = FD*sum(v_p1);
42 payoff_p2(t) = FD*sum(v_p2);
43 end
44
45 figure(1)
46 scatter(A1,B1, 'filled')
47 hold on
48 plot(payoff_p1,payoff_p2,'o','MarkerSize',1,...
49          'MarkerFaceColor',[0.5 0.5 0.5],'MarkerEdgeColor', [0.5 0.5 0.5])
50 xlabel('Player 1')
51 ylabel('Player 2')
52 title(['Type 3 FD Game Reward Space. T = ' num2str(T) ' iterations'])
53 hold off
```

D Algorithm 4: Q Checker

```
%loop to match px and x*. insert in type 3 algorithm
%---[1] Frequency vector x (scatter)
%  for i = 1:8
%      r(i) = betarnd(0.5,0.5);
% end
% for i = 1:8
%      x(i) = r(i)/(sum(r));
% end
% %----
%---[1] Frequency vector x (outline)
for i= 1:3
    r(i) = betarnd(0.5,0.5);
end
Norm_val = sum(r);
for i = 1:3 %
    x_a(i) = r(i)/Norm_val;
end
x_b1 = [x_a(1) x_a(3) 0 0]';%
x_b2 = [x_a(2) 0 0 0]';%
x_c1 = x_b1(randperm(length(x_b1))); %randomises positions of x_b
x_c2 = x_b2(randperm(length(x_b2)));
for i =1:8
    if i< 5
        x(i) = x_c1(i);
    else
        x(i) = x_c2(i-4);
    end
end
% %----
%---[2] Intermediate y vector, Q and x*
for i = 1:4
    y(i) = x(i)/(sum(x(1:4)));
end
for i =5:8
    y(i) = x(i)/(sum(x(5:8)));
end
```

```
px = p -  x*matrixA;
for w = 1:4
for i = 1:4
    yp(i) = y(i+4)*px(i+4);
    yp_not(i) = y(i)*(1-px(i));
end
Q =( sum(yp)) / ((sum(yp_not)) + (sum(yp)));
Q_not = 1-Q;
Q_vec(w) = Q;
for i =1:4
    xstar(i) = Q*y(i);
end
for i = 5:8
    xstar(i) = Q_not*y(i);
end
px = p -  xstar*matrixA;
end
Q_check_1 = Q_vec(1)- ((Q_vec(2)-Q_vec(1))^2)/(Q_vec(3)- 2*Q_vec(2) + Q_vec(1)
    );
Q_check_2 = Q_vec(2)- ((Q_vec(3)-Q_vec(2))^2)/(Q_vec(4) - 2*Q_vec(3) + Q_vec
    (2));
diff = Q_check_1 - Q_check_2;

while diff > abs(1e-8)

    Q_check_1 = Q_check_2;
for i = 1:4
    yp(i) = y(i+4)*px(i+4);
    yp_not(i) = y(i)*(1-px(i));
end
Q =( sum(yp)) / ((sum(yp_not)) + (sum(yp)));
Q_not = 1-Q;
Q_vec(w+1) = Q;
for i =1:4
    xstar(i) = Q*y(i);
end
for i = 5:8
```

```
72      xstar(i) = Q_not*y(i);
73  end
74  px = p -  xstar*matrixA;
75  Q_check_2 = Q_vec(w-1)...
76  - ((Q_vec(w)-Q_vec(w-1))^2)/(Q_vec(w+1) - 2*Q_vec(w) + Q_vec(w-1));
77  diff = Q_check_1 - Q_check_2;
78  w = w+1;
79  end
```