Master Thesis

Stationary (ϵ-)equilibria in 2- and 3-player quitting games

L.C.A. Meessen

Master Thesis DKE 12-02

Thesis submitted in partial fulfillment
of the requirements for the degree of Master of Science
of Operations Research at the Department of Knowledge
Engineering of the Maastricht University

Thesis Committee:

Dr. G. Schoenmakers
Dr. J. Derks

Maastricht University
Faculty of Humanities and Sciences
Department of Knowledge Engineering
Master Operations Research

March 23, 2012

Abstract

We construct a program in Matlab to find all stationary (ϵ-)equilibria in 2-player and 3-player quitting games. These games are n-player sequential games in which at any stage, each player has the choice between continuing and quitting. As soon as a (set of) player(s) decides to quit, the game ends. The payoffs the players receive depend only on the set of players who actually quit.
Our program is able to find all (ϵ-)equilibria in 2-player quitting games. For 3-player quitting games, our program is able to provide the stationary equilibria in a game or to show that a game does not have a stationary equilibrium. We also calculate the probability that a random 3-player quitting game has a stationary equilibrium. In cyclicly symmetric random games the probability of finding stationary equilibria turns out to be higher than in normal random games.

Contents

Chapter 1

Introduction

The rules of a quitting game are very simple. The players can choose to play either continue or quit. As long as no player quits, the game continues. As soon as a (set of) player(s) decides to quit, the game ends. The attraction of quitting games lies in the fact that although the games seem very simple, playing the game optimally can be very hard.
The definition of a quitting game was given in 2001 by Solan and Vieille [9]:

> Quitting games are n-player sequential games in which, at any stage, each player has the choice between continuing and quitting. The game ends as soon as at least one player chooses to quit; player i then receives a payoff r_S^i, which depends on the set S of players that did choose to quit. If the game never ends, the payoff to each player is 0.

A function $x^i = (x_n^i) : \mathbb{N} \to [0,1]$ that gives the probability that player i quits at stage n, is called a strategy. This simple function is a strategy because the strategic behaviour in a quitting game does not depend on the history of the play: the game stays in the same state as long as no player quits. If the game gets a history, the game ends because then a (set of) player(s) has decided to quit. A strategy is called stationary if $x_n^i = x_1^i \ \forall n \in \mathbb{N}$. This means that in every stage of the game, player i always plays the same randomization thus x_n^i is independent of n. A strategy is called pure if $x_n^i \in \{0,1\} \ \forall n \in \mathbb{N}$. A vector with strategies, one for each player, is called a profile $\mathbf{x} = (x^i)_{i \in N}$ with N the set of players.
When playing the game, the players get a payoff. For quitting games the payoff for player i, when profile $\mathbf{x}$ is played, is given by the following formula:

$$\gamma^i(\mathbf{x}) = \begin{cases} 0 & \text{if each player always selects continue} \\ r_S^i & \text{if set } S \text{ quits} \end{cases}$$

Each player wants to maximize the payoff he is getting. For players to maximize their payoff, they use the following observation: For every set of stationary

strategies of the other players there is a pure stationary best reply for a player. A stationary best reply is an strategy for which a player does the best he can against what his opponents did.
An equilibrium is a profile that are best replies against each other. A set of strategies is an ϵ-equilibrium if neither player can gain more than ϵ by deviation, thus if for every player i and every strategy $\mathbf{y}^i$ of player i,

$$\gamma^i(\mathbf{x}^i, \mathbf{x}^{-i}) \geq \gamma^i(\mathbf{y}^i, \mathbf{x}^{-i}) - \epsilon \quad \forall \mathbf{y}^i$$

The notation $\mathbf{x}^{-i}$ is used to denote a profile where player i is not included. So $\mathbf{x}^i$ is a best reply against $\mathbf{x}^{-i}$ $\forall i$. A 0-equilibrium is simply called an equilibrium. It is difficult to find an equilibrium in a quitting game for $n \geq 3$ players, as example 2 will show.
Quitting games are subsets of more general games, for which the existence of equilibria is proven. For example, a quitting game is a special kind of recursive repeated game with absorbing states. A state in a game is absorbing, if the state is never left after entering. In 1957 Everett introduced recursive games [2]. Recursive games are games in which the non-zero payoffs are only received when the games ends. The game ends as an absorbing state is reached. If the game never reaches an absorbing state, the payoff is 0. A repeated game with absorbing states, also called an absorbing game, is a sequential game with only one nonabsorbing state. For a quitting game, the game stays in the nonabsorbing state with zero payoff as long as no player quits. As soon as a (set of) player(s) quits, the game moves to an absorbing state and the players get the corresponding payoff.
When considering a quitting game as a special kind of repeated game with absorbing states, the first interesting result with respect to equilibria is found. In 1989 it was proven by Vrieze and Thuijsman [11] that ϵ-equilibria exist in 2-player repeated games with absorbing states. These equilibria are not always stationary, an example is the game called 'the big match' introduced in 1957 by Gillette [6]. In 1968, Blackwell and Ferguson [1] proved the existence of an ϵ-equilibrium for this game, but the strategies are not stationary because they depend on the history of the game. Since quitting games are a special kind of a recursive repeated game with absorbing states, we conclude that for 2-player quitting games ϵ-equilibria exist. Solan and Vieille [10] showed that stationary (ϵ-)equilibria exist in 2-player games.
In the next example we will demonstrate the existence of the pure equilibrium (continue, continue) and the application of γ^i for finding the equilibrium. We will always write quitting games in this way: the first action is 'continue' and the second action is 'quit'.

Example 1:
Consider the 2-player quitting game:

		2 c	**2** q
1	c		2,-2
	q	-2,2	1,1

The equilibrium in this example consists of the pure stationary strategies $x^1 = 0$ and $x^2 = 0$, meaning that both players always choose the action continue. The best reply for player 2 is always to play continue no matter what player 1 does, since

1. if $x^1 = 0$, then 'continue' yields a payoff of 0 while 'quit' yields a payoff of -2. So continue is the best reply for player 2 to player 1 playing continue,
2. if $x^1 > 0$, then 'continue' yields a payoff of 2, while 'quit' yields a payoff of $-2(1 - x^1) + 1 \cdot x^1 < 2$. So continue is also the best reply for player 2 to an action other than continue of player 1.

The same reasoning can be applied for player 1.
Another way to see why (continue, continue) is an equilibrium, look at the expected payoff the players are getting. The formula for the expected payoff is:

$$\gamma(\mathbf{x}) = (1-x^1)x^2(2,-2) + x^1(1-x^2)(-2,2) + x^1x^2(1,1) + (1-x^1)(1-x^2)\gamma(\mathbf{x})$$

In the formula, the probabilities are multiplied by the payoff in the corresponding matrix entry. Only for the action pair (continue, continue), meaning $x^1 = 0$ and $x^2 = 0$, the players get a zero payoff and have to play the game again. In the formula this shows by $(1 - x^1)(1 - x^2)\gamma(\mathbf{x})$, because we expect the players to get the same payoffs if they play the game again. We can rewrite the formula for the expected payoff in another way:

$$\gamma(\mathbf{x}) = \frac{(1-x^1)x^2(2,-2) + x^1(1-x^2)(-2,2) + x^1x^2(1,1)}{1-(1-x^1)(1-x^2)} \tag{1.1}$$

We can use formula (1.1) to show that (continue, continue) is an equilibrium. If player 2 plays $x^2 = 0$, then

$$\begin{array}{ll} \gamma^1(\mathbf{x}) = \frac{x^1(-2)}{1-(1-x^1)} = -2 & \text{if } x^1 > 0 \\ \gamma^1(\mathbf{x}) = 0 & \text{if } x^1 = 0 \end{array}$$

and thus the expected payoff of player 1 is going to be negative if he does not continue; deviation from player 1 is only profitable for player 2. The same reasoning is true for player 2.

The game in the previous example was symmetric. The definition of a symmetric game is the following:

Definition 1:
A symmetric game is a game in which the identity of a player does not change the resulting game. Each player earns the same payoff when making the same choice against similar choices of his competitors. So for two players a symmetric game looks like:

		2
1		a_2, a_1
	a_1, a_2	a_3, a_3

where the first payoff is for player 1 and the second payoff for player 2. For each n-player symmetric game the game does not change even if you rename the players.

In general, for a two player game with payoffs a_1, a_2, a_3 for player 1 and payoffs b_1, b_2, b_3 for player 2,

		2
1		a_2, b_2
	a_1, b_1	a_3, b_3

(1.2)

the expected payoff for player 1 is:

$$\gamma(\mathbf{x}^1) = \frac{(1-x^1)x^2a_2 + x^1(1-x^2)a_1 + x^1x^2a_3}{1-(1-x^1)(1-x^2)} \tag{1.3}$$

Now to the 3-player case. We look at an example with a completely mixed stationary equilibrium. A completely mixed stationary equilibrium is an equilibrium where all players play a randomization of their actions. The game in the example is a cyclicly symmetric game, therefore we will first define such a game.

Definition 2:
A cyclicly symmetric game is where the payoffs and the absorbing entries are cyclicly symmetric. This means for three players that

$$r^1(a_i, a_j, a_k) = r^2(a_k, a_i, a_j) = r^3(a_j, a_k, a_i)$$

for each entry (a_i, a_j, a_k). A cyclicly symmetric game looks as follows, for three players:

		3		
		2		**2**
1		a_3, a_1, a_2	a_2, a_3, a_1	a_6, a_4, a_5
	a_1, a_2, a_3	a_4, a_5, a_6	a_5, a_6, a_4	a_7, a_7, a_7

For two players a cyclicly symmetric game and a symmetric game look the same.

Example 2:
Consider the cyclicly symmetric 3-player game:

3

		2		**2**
1		0,4,1	1,0,4	4,1,0
	4,1,0	1,0,4	0,4,1	-1,-1,-1

The completely mixed stationary equilibrium in this game is:

$$\text{player 1:} \quad x^1 = \sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}} - \frac{5}{36\sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}}} + \frac{11}{6} \approx 0.4178,$$

$$\text{player 2:} \quad x^2 = \sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}} - \frac{5}{36\sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}}} + \frac{11}{6} \approx 0.4178,$$

$$\text{player 3:} \quad x^3 = \sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}} - \frac{5}{36\sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}}} + \frac{11}{6} \approx 0.4178$$

For a completely mixed stationary equilibrium to exist we need that: player 1 is indifferent if player 2 plays x^2 and player 3 plays x^3; player 2 is indifferent if player 1 plays x^1 and player 3 plays x^3 and player 3 is indifferent if player 1 plays x^1 and player 2 plays x^2. By symmetry we have

$$(x^1, x^2, x^3) = (x^2, x^3, x^1) = (x^3, x^1, x^2)$$

and thus for the completely mixed equilibrium where all players are indifferent $x^1 = x^2 = x^3$. In a cyclicly symmetric game, if there exist a completely mixed equilibrium it is a symmetric equilibrium.
Assume that player 1 and 2 both quit with probability x, now we want to find the value for x for which player 3 is indifferent. When player 3 plays quit, that yields a payoff of:

$$r(q) = -1 \cdot x^2 + 4(1-x)^2 + 1 \cdot x(1-x) = 2x^2 - 7x + 4$$

When player 3 plays continue, he gets a payoff of:

$$r(c) = \frac{4x^2 + 1 \cdot x(1-x)}{1-(1-x)^2} = \frac{3x^2 + x}{2x - x^2} = \frac{3x+1}{2-x}$$

Player 3 is indifferent if $r(q) = r(c)$, thus

$$\begin{aligned} & 2x^2 - 7x + 4 = \tfrac{3x+1}{2-x} \\ \Rightarrow \quad & (2x^2 - 7x + 4)(2-x) = 3x + 1 \\ \Rightarrow \quad & 2x^3 - 11x^2 + 21x - 7 = 0 \end{aligned}$$

Solving $2x^3 - 11x^2 + 21x - 7 = 0$ for x we get:

$$x = \sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}} - \frac{5}{36\sqrt[3]{\frac{\sqrt{60900}}{144} - \frac{185}{108}}} + \frac{11}{6}$$

which is exactly the solution for the completely mixed stationary equilibrium.

Not all 3-player quitting games have a stationary equilibrium. Flesch, Thuijsman and Vrieze [5, 4] investigate the following 3-player quitting game:

Example 3:
Consider the cyclicly symmetric 3-player game:

3

		2		**2**
1		0,1,3	3,0,1	1,1,0
	1,3,0	1,0,1	0,1,1	0,0,0

In 1996, Flesch, Thuijsman and Vrieze showed that in this game no stationary ϵ-equilibrium exists, this proves that the two-player case does not extend to the n-player case. However, the game admits a cyclic equilibrium, where the players put in turns a positive weight on the quitting action and play continue the rest of the cycle. The simplest equilibrium strategies in this game are:

player 1: $\mathbf{x}^1 = (\frac{1}{2}, 0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, \ldots)$,
player 2: $\mathbf{x}^2 = (0, \frac{1}{2}, 0, 0, \frac{1}{2}, 0, 0, \ldots)$,
player 3: $\mathbf{x}^3 = (0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, 0, \ldots)$

Playing these equilibrium strategies, the players get an expected payoff

$$\begin{aligned} \gamma(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3) &= [\tfrac{1}{2}(1,3,0) + (\tfrac{1}{2})^2(0,1,3) + (\tfrac{1}{2})^3(3,0,1)] \textstyle\sum_{t=0}^{\infty} (\frac{1}{2})^{3t} \\ &= (1,2,1) \end{aligned}$$

The formula for the expected payoff can be explained in the following way:

- if player 1 quits at stage 1, which happen with probability $\frac{1}{2}$, the expected payoff is: $(1,3,0)$,
- if player 1 did not quit and player 2 quits at stage 2, which happens with probability $\frac{1}{2} \cdot \frac{1}{2}$, then the expected payoff is: $(0,1,3)$,
- if player 1 and 2 did not quit and player 3 quits at stage 3, which happens with probability $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$, the expected payoff is: $(3,0,1)$,
- the probability that none of the players quits in the previous time steps and the game continues: $(\frac{1}{2})^3$ and then the game essentially starts once again.

In 1999 Solan proved that for 3-player absorbing games equilibria exist [8]. So in 3-player quitting games equilibria exist. The simplest equilibria can either be stationary or of a cyclic nature. Similar to the 3-player game in the previous example, Solan and Vieille [10] have proven that the following 4-player game has an equilibrium of a cyclic nature as well.

Example 4:
Consider the 4-player game:

4

3		**2**		**2**
1		4,1,0,0	**1**	0,0,4,1 | 1,1,0,1
	1,4,0,0	1,1,1,1		1,0,1,1 | 0,1,0,0
1	0,0,1,4	0,1,1,1	**1**	1,1,1,1 | 0,0,1,0
	1,1,1,0	1,0,0,0		0,0,0,1 | -1,-1,-1,-1

Solan and Vieille proved that the game admits a cyclic equilibrium profile $\mathbf{y}$ with period 2 and the following structure:

$$y_n = \begin{cases} (x, 0, z, 0) & n \text{ odd} \\ (0, x, 0, z) & n \text{ even} \end{cases}$$

where $x, z \in]0, 1[$ are independent of n, that is, at odd stages players 2 and 4 continue, while 1 and 3 quit with positive probability, whereas at even stages 1 and 3 continue, while 2 and 4 quit with positive probability.

Games that are (closely) related to quitting games are free transition games. In a quitting game the players act simultaneously, while in a free transition game, the players act one at a time. In a free transition game at each stage one player is in control of the game.
In a free transition game, the player in control of the game is called the active player. Each player is the active player in only one state of the game. The active player i has the choice to terminate the game, giving each player a payoff corresponding the payoff vector of player i, or to announce a new active player. If a new active player is announced, the players receive no payoff and the game continues. A free transition game and a quitting game have in common that as long as no player chooses to quit, the game continues and the players receive no payoff. A free transition game is comparable with a quitting game where players get a worse payoff if more than one player quits at a time.
A history h of a free transition game is a finite sequence of players. This sequence is the list of active players in chronological order. Unlike a quitting game, the strategies in a free transition game can be history dependent. An (ϵ-)equilibrium is called subgame-perfect if for any finite history h the strategies prescribe to play an (ϵ-)equilibrium after h.
For free transition games, some interesting results are found. In 2009, Kuipers et al. proved the existence of subgame-perfect equilibria in free transition games with payoffs ≥ 0 [7]. For free transition games where negative payoffs are allowed, subgame-perfect equilibria exist as Flesch et al. proved in 2011 [3].
Now returning to the quitting games which form the basis of this paper. To find the equilibria in quitting games, we need a lot of calculations. The question is whether a computer program can help finding these equilibria much faster and easier. The problem statement of this report is based on the previous

question, namely: 'Using Matlab to find stationary (ϵ-)equilibria in 2- and 3-player quitting games'. The idea is that a computer program can help solving difficult calculations for finding the equilibria like in example 2.
For finding an answer to the problem statement, some research questions are used to split the problem into smaller pieces. The research questions are:

- Is it possible to set up an algorithm for finding stationary (ϵ-)equilibria in 2-player quitting games?
- Is it possible to set up an algorithm that correctly identifies a stationary equilibrium in a 3-player quitting game, or that says that there exists no stationary equilibrium?
- Can an algorithm prove the existence of equilibria in 3-player quitting games?

In chapter 2 there will be a description of some algorithms used to find stationary equilibria in quitting games or to show that such equilibria cannot be found in specific quitting games. In chapter 3 experiments are discussed and the concluding chapter describes an answer to the research questions and problem statement in the form of a conclusion with some advice for further research.

Chapter 2

Algorithms

To find stationary equilibria, we constructed two algorithms: one to find all stationary equilibria in 2-player games and one to find a stationary equilibrium in 3-player games or to show that such equilibria do not exist. First the algorithm for two players is given and then the algorithm for the games with 3 players.

2.1 Finding stationary equilibria for two players

The algorithm for finding stationary equilibria for two players uses the principle that for every stationary strategy of one player there is a stationary best reply for the other player. The basic idea for finding the equilibria with this algorithm is just to look at all pairs where one player plays a stationary strategy and the other player plays a stationary best reply against this strategy. So for both players we made a list consisting of the stationary strategies of his opponent and all of his best replies to such a strategy. For finding the equilibria of the game, just look at the strategy pairs that are best replies to each other.
The input for the main function, called `twoplayers`, is the game matrix. The game matrix used as input for `twoplayers` looks similar to the matrix in equation (1.2) in chapter 1 but with some adaptions. The input for `twoplayers` is the matrix:

$$\begin{bmatrix} 0 & 0 & a_2 & b_2 \\ a_1 & b_1 & a_3 & b_3 \end{bmatrix} \tag{2.1}$$

Here, the zeros are the payoffs for the action pair (continue, continue) and the odd columns contain the payoffs for player 1, while the even columns give the payoffs for player 2. The function `twoplayers` first splits the game in equation (2.1) into two separate matrices, one with the payoffs for player 1 and the other with the payoffs for player 2. The game can be split, because for finding the best reply of player i we only look at the payoffs for player i. The matrix with the payoffs for player 1 is transposed, since then we only need one function for finding the best reply. The matrices used as input for the function `findreply2p`

look like:

For player 1:

	c^2	q^2
c^1		a_2
q^1	a_1	a_3

represented as:

	c^1	q^1
c^2		a_1
q^2	a_2	a_3

and For player 2:

	c^2	q^2
c^1		b_2
q^1	b_1	b_3

Since in the function for finding the reply, there is no difference made between the two players, the matrix used as input for finding the best reply looks like:

	$1-y$	y
$1-x$	0	b
x	c	d

The function `findreply2p` considers x as the stationary strategy for one player and y is seen as the reply of the opponent. In this case, x and y give the probability that a player chooses to quit. If the strategy of the player is mixed, then the function for finding the reply first looks at the case for which the opponent is indifferent. For this, the value of x is found by solving:

$$c = xd + (1-x)b$$

for x. This yields:

$$x = \frac{b-c}{b-d} \tag{2.2}$$

Rewriting equation (2.2) using the original game for player 1 we get:

$$z_1 = \frac{b_2 - b_1}{b_2 - b_3} \tag{2.3}$$

and for player 2:

$$z_2 = \frac{a_1 - a_2}{a_1 - a_3} \tag{2.4}$$

For $0 < z_1, z_2 < 1$, we have that (z_1, z_2) is a stationary equilibrium. The function `findreply2p` also finds the best reply for the opponent if the player plays a strategy that differs from x. All the functions used for this algorithm use the value ϵ to point out a small difference which is close to zero, but not equal to zero.

After looking at all the stationary strategies of a player and finding the opponent's best reply, a list is created with strategy pairs for both players. When the lists with intervals are created, the function for finding the equilibria looks if there are overlapping intervals in the two lists. The application of the algorithm is best explained by means of an example.

Example 5:
Consider the following two player game:

	2,4
0,3	3,1

After calling the function `findreply2p` and finding best replies for a player to his opponent's strategies the following two lists are created:

player 1

strategy pl 1		best reply pl 2	
0	0	ϵ	1
1	1	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	1
ϵ	$\frac{1}{3}-\epsilon$	1	1
$\frac{1}{3}+\epsilon$	$1-\epsilon$	0	0

player2

best reply pl 1		strategy pl 2	
0	1	0	0
1	1	1	1
0	1	$\frac{2}{3}$	$\frac{2}{3}$
0	0	ϵ	$\frac{2}{3}-\epsilon$
1	1	$\frac{2}{3}+\epsilon$	$1-\epsilon$

The values $\frac{1}{3}$ for player 1 and $\frac{2}{3}$ for player 2 can be verified using formulas (2.3) and (2.4):

$$z_1 = \frac{4-3}{4-1} = \frac{1}{3}$$

and,

$$z_2 = \frac{0-2}{0-3} = \frac{2}{3}$$

The above lists contain the intervals for the strategies and best replies as output given by Matlab. For easier reading we will rewrite the lists and during the rest of this paper we will use the following representation for strategies, best replies and equilibria:

player 1

strategy pl 1	best reply pl 2
0	(0,1]
1	0
$\frac{1}{3}$	[0,1]
$(0, \frac{1}{3})$	1
$(\frac{1}{3},1)$	0

player2

best reply pl 1	strategy pl 2
[0,1]	0
1	1
[0,1]	$\frac{2}{3}$
0	$(0,\frac{2}{3})$
1	$(\frac{2}{3},1)$

For finding the equilibrium of the game, look at strategy pairs that both players have in common or where there is an overlap in the strategy pairs of both players. The function `findeq2p` does compare the lists with strategy pairs. When looking for the equilibrium intervals, there can be really long tables with all possible equilibria. For getting a more compact table, look at the intervals that are adjacent for a fixed strategy of the other player. Thus, look if it is possible to combine some equilibrium intervals into a bigger equilibrium interval, which is done by the function `findadjacent`. The output for the function `twoplayers` are the following equilibrium intervals for this example:

strategy pl 1	strategy pl 2
0	$(0, \frac{2}{3}]$
$[\frac{1}{3},1]$	0
$\frac{1}{3}$	$\frac{2}{3}$

The Matlab code of the functions for finding the equilibria in a 2-player quitting game can be found in appendix A.1. There is an overview of all functions used for the algorithm and how main program `twoplayers` uses the functions `findreply2p` and `findeq2p`.

2.2 Finding stationary equilibria for 3 players

The algorithm described in section 2.1 gets really complicated if there are more than two players. Just looking at the difference between a 2-player and a 3-player quitting already reveals that the 3-player game is more complex. Adding a player to a quitting game makes the strategy combinations grow exponentially. A 3-player quitting game looks like:

3

	2	
1		a_2, b_2, c_2
	a_1, b_1, c_1	a_3, b_3, c_3

	2	
1	a_4, b_4, c_4	a_6, b_6, c_6
	a_5, b_5, c_5	a_7, b_7, c_7

(2.5)

In this game, a_i, b_i and c_i are the payoffs for player 1, player 2 and player 3. For the algorithm in this section, the original game matrix in equation (2.5) is rewritten in the form:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{bmatrix} \tag{2.6}$$

The payoff r_S^i is poisitioned at location a_{ik} in the matrix A with $k = \sum_{j \in S} 2^{j-1}$ (with the players labeled 1,2,...).
For the algorithm the players each belong to one of the following groups:

- S_i: the group of players that is indifferent
- S_c: the group of players that plays continue
- S_q: the group of players that plays quit

If a player is indifferent, then his expected payoff for playing continue is equal to the expected payoff for playing quit. If a player plays continue or quit then his expected payoff for this action is as least as good as his expected payoff for the other action. Thus in formula it looks like:

$$\begin{array}{ll} \forall j \in S_i & \gamma_j(\mathbf{x}^{-j}, 1) = \gamma_j(\mathbf{x}^{-j}, 0) \\ \forall j \in S_c & \gamma_j(\mathbf{x}^{-j}, 1) \geq \gamma_j(\mathbf{x}^{-j}, 0) \\ \forall j \in S_q & \gamma_j(\mathbf{x}^{-j}, 1) \leq \gamma_j(\mathbf{x}^{-j}, 0) \end{array} \tag{2.7}$$

The expected payoff γ_i is built as follows:

$$\gamma_i(\mathbf{x}) = c(\mathbf{x})(A\overline{\mathbf{x}}) \tag{2.8}$$

where A is the game matrix as represented in equation (2.6), $\overline{\mathbf{x}}$ is a vector with products corresponding to probabilities per column and $c(\mathbf{x})$ is a formula to correct that the action continue has a probability not equal to 1. A better way to explain this is to rewrite the expected payoff γ_i as

$$\gamma_i(\mathbf{x}) = A\overline{\mathbf{x}} + p(\mathbf{x})\gamma_i(\mathbf{x})$$

with $p(\mathbf{x})$ the probability that all players play continue, i.e., $p(\mathbf{x}) = \prod_{i=1}^{n}(1-x^i)$. For $c(\mathbf{x})$ we then get,

$$c(\mathbf{x}) = \frac{1}{1 - \prod_{i=1}^{n}(1-x^i)} \tag{2.9}$$

where n is the number of players playing the game. For the 2-player case we have

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{2.10}$$

$$\overline{\mathbf{x}} = \begin{bmatrix} x^1(1-x^2) \\ (1-x^1)x^2 \\ x^1x^2 \end{bmatrix}$$

and

$$c(\mathbf{x}) = \frac{1}{1-(1-x^1)(1-x^2)}$$

Notice that in equation (1.3), the formula for the expected payoff in section 1, the denominator is equal to the denominator in equation (2.9). Also the numerator in equation (1.3) is equal to $A\overline{\mathbf{x}}$. Using the information of equation (2.7), the function `gsolve` can than be used to find stationary equilibria in n-player quitting games if they exist.
For each player that is indifferent, an equation is set up where the expected payoff of playing continue is equal to the expected payoff of playing quit. The equations are rewritten in such a way that we have:

$$\gamma_j(\mathbf{x}^{-j}, 1) - \gamma_j(\mathbf{x}^{-j}, 0) = 0$$

The Matlab function 'solve' is called, for finding values for the symbolic variables x_i in these equations. The vector x only contains symbolic variables for the players that are indifferent. For the players that choose to continue, the symbolic variable x is replaced by a zero and for the players that quit a 1 is substituted instead of the symbolic variable. The function `gsolve` returns a vector with the numeric solutions for the symbolic variables, if they exist.
The source code for the algorithm can be found in the appendix A.2.

Chapter 3

Experiments and results

In this chapter, the algorithms of chapter 2 are tested with some games and from the results of these tests a conclusion is given later in chapter 4. The tests with the different algorithms are discussed per algorithm with some specific examples. First the algorithms are tested with 2-player quitting games, later the algorithm for 3 players is also tested with 3-player quitting games.

3.1 Testing the two-player method

The algorithm discussed in section 2.1 is tested here with several interesting examples. It should be kept in mind that x^1 and x^2 give the probability with which a player decides to quit. First there will be some examples of games with non-negative payoffs, later there will also be some games with a negative payoff for one or both players. If in the equilibria the value ϵ appears, then this means a very small value close to zero but not equal to zero. All the answers for the equilibria are checked by doing the calculations manually, but for convience only the equilibria are provided and not the complete calculations to check them. We start with an easy example, where 'everything' is an equilibrium.

Example 6:
The first example is the game where all payoffs are equal to zero, thus

	0,0
0,0	0,0

The function `twoplayers` finds:

x^1	x^2
[0,1]	[0,1]

This means that every action combination is an equilibrium. No matter what action combination is played the players always receive zero.

Example 7:
Another interesting game is a game where all payoffs in the absorbing states are equal to 1, thus

	1,1
1,1	1,1

Here obviously 'everything' except always (continue, continue) is an equilibrium. The function `twoplayers` finds:

x^1	x^2
[0,1]	(0,1]
(0,1]	[0,1]

After those easy games, we will now provide a few slightly more complex games.

Example 8:
Consider the symmetric game:

	2,1
1,2	1,1

In this game, both players like to play the continue action and hope that the other player will quit. The equilibria for this game are:

x^1	x^2
0	(0,1]
(0,1]	0

These equilibria are logical when looking at the game. If both players play continue, then playing quit is better because some payoff is better than receiving nothing so this cannot be an equilibrium. Playing (x^1, x^2) with $x^1 > 0$ and $x^2 > 0$ is also not an equilibrium because one of the players can easily deviate to playing continue and receiving a higher payoff.

Example 9:
Now lets look at a game where player 2 has some negative payoffs and player 1 still has only positive payoffs:

	2,-2
0,-1	3,1

The equilibria in this game are:

x^1	x^2	**expected payoff**
$[0,\frac{1}{3}]$	0	(0,-1)
$\frac{1}{3}$	$\frac{2}{3}$	(2,-1)
1	1	(3,1)

The expected payoffs are calculated using formula (1.3). Playing (quit,quit) is the most liked equilibrium, because then the players both receive the highest possible payoff. Playing anything else than (quit,quit) or (continue, continue) gives a probability that player 2 has a negative payoff but player 2 cannot play a strategy to avoid ending up with a negative payoff.

Example 10:
Now another game where each player has one negative payoff:

	2,-1
1,1	-1,2

This game has the equilibria:

x^1	x^2
$(0, \frac{2}{3}]$	0

By playing continue all the time player 2 can ensure that he does not get a negative payoff, because getting nothing is still better than paying something. Player 1 does not play continue or quit as a stationary strategy, because here player 2 can easily ensure that player 1 gets no payoff or a negative payoff. For $x^1 = \frac{2}{3}$ player 2 is indifferent and player 1 puts some more weigth on the action continue since there he gets a positive payoff. If player 1 plays $x^1 > \frac{2}{3}$, then $x^2 = 1$ is the best reply for player 2, leading to the -1 payoff for player 1 with a high probability.

Example 11:
In this game both players have one positive payoff and two negative payoffs:

	2,-2
-1,-1	-3,1

The only equilibrium in this game is:

x^1	x^2
0	0

Player 1 avoids playing the quit action because there are only negative payoffs for him. Knowing that player 1 does not choose quit, it is obvious for player 2 to also choose continue and not having to pay anything.

Example 12:
This is a symmetric game where both players have two negative payoffs:

	-2,2
2,-2	-3,-3

The symmetry of the game, is also showing in the equilibria:

x^1	x^2	**expected payoff**
0	$[\frac{4}{5},1]$	(-2,2)
$[\frac{4}{5},1]$	0	(2,-2)
$\frac{4}{5}$	$\frac{4}{5}$	(-2,-2)

If a player plays the stationary strategy where the probability of quitting is $\frac{4}{5}$, then the other player will be indifferent. Also if a player chooses to continue all the time, the best reply for the opponent is to quit with a positive probability because then he receives a positive payoff.

Example 13:
This last example is a game which contains only negative payoffs for both players and is also symmetric:

	-2,-2
-2,-2	-1,-1

The only two equilibria in this game consist of pure strategies for the players,

namely:

x^1	x^2
0	0
1	1

For both players it is worst if they play (0, x^2) or (x^1, 0), for $x^1 > 0$ and $x^2 > 0$. The two choices that remain are (continue, continue) were the payoff is 0 and (quit, quit) were both players have to pay 1.

3.2 The validation of the function `gsolve` for 2-player games

The function `gsolve` is tested based on some theorems about 2-player quitting games. Below the theorems and the proofs are stated and after that the test results to confirm the validity of the function `gsolve`.

Theorem 1. *In 2-player quitting games with non-negative payoffs pure stationary equilibria exist.*

Proof. We have a quitting game

	a_2, b_2
a_1, b_1	a_3, b_3

(3.1)

with payoffs a_1, a_2, a_3 for player 1 and b_1, b_2, b_3 for player 2 all being non-negative. The pure stationary strategy pair (continue, quit) is a pure equilibrium if $a_2 \geq a_3$ because neither player has a profitable deviation. Similarly, if $b_1 \geq b_3$ then the pure stationary strategy pair (quit, continue) is a pure equilibrium. If neither of the previous mentioned stationary strategy pairs leads to a pure equilibrium, then the stationary strategy pair (quit, quit) will be a pure equilibrium. □

Theorem 1 does not apply to 2-player quitting games in general which is demonstrated in the next example.

Example 14:
Consider the following game:

	-1,1
-1,1	1,-1

In this game, no pure stationary equilibrium exists because in every pure stationary strategy pair a player can profit by deviating namely:

- (continue, continue), player 2 can profit by deviating and playing quit. Then he gets a payoff of 1 instead of 0.

- (continue, quit), if player 1 quits he gets 1 instead of -1 which is better for him.
- (quit, quit), now player 2 can get 1 by deviating and playing continue
- (quit, continue), player 1 profits by playing continue because getting nothing for him is always better than to pay 1.

The function `gsolve` can be used to find pure stationary equilibria but it is more useful in finding mixed stationary equilibria. The next theorem will give an important result for the validation of the function `gsolve`.

Theorem 2. *The probability that a random 2-player quitting game has a mixed stationary equilibrium, is* $\frac{1}{9}$.

Proof. We have the same game as in equation (3.1), with as only difference that the payoffs do not have to be positive but can have any random value. Notice that the probability that two random numbers drawn from a continuous distribution are equal is zero. This means that six orderings are possible for the payoffs of player 1, all of which are equally likely, namely: $a_1 < a_2 < a_3$, $a_1 < a_3 < a_2$, $a_2 < a_1 < a_3$, $a_2 < a_3 < a_1$, $a_3 < a_1 < a_2$ and $a_3 < a_2 < a_1$. The following situations can occur:

- $a_2 > \max(a_1, a_3)$ for which player 1 always plays continue (this happens with probability $\frac{1}{3}$),
- $a_2 < \min(a_1, a_3)$ for which player 1 always quits (also with probability $\frac{1}{3}$),
- $a_1 < a_2 < a_3$ or $a_3 < a_2 < a_1$ with probability $\frac{1}{3}$. Here player 2 can play a randomization that makes player 1 indifferent.

Following the similar reasoning for player 2, if $b_2 < b_1 < b_3$ or $b_3 < b_1 < b_2$ then player 1 can play a randomization that makes player 2 indifferent. The probability that this occurs is $\frac{1}{3}$. So, in a random 2-player quitting game, the probability that a completely mixed equilibrium exists, is $\frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$. □

The function `gsolve` is tested with 2-player games, based on finding stationary strategies for which both players are indifferent. The function `findx2p` is used to validate the function `gsolve`. The function `findx2p`, which can be found in appendix A.3, uses formula (2.2) to find the probability for which a player is indifferent. The game input for the algorithms is formula (2.10) in section 2.2. In both tests, we run a simulation with 10000 random games with either positive payoffs only or payoffs between -1 and 1. The function 'rng' is a built-in Matlab function, that seeds the random number generator 'rand' to produce a reproducible sequence of random numbers. We count the number of times a mixed equilibrium is found, which we call check. And for comparison of `gsolve` and `findx2p` we also look at the time in seconds it takes to perform the simulation. Since we know that a mixed stationary equilibrium exist in a random game with probability $\frac{1}{9}$, performing a simulation with 10000 random games should give us a value for check which is around $10000 \cdot \frac{1}{9} \approx 1111$.

Table 3.1: Results for the simulations with a specific starting seed and 10000 random games for the functions `findx2p` and `gsolve`. There are two tables, 3.1(a) for random games with only positive payoffs and 3.1(b) for random games with payoffs between -1 and 1.

(a) Random games with positive payoffs

seed	function	check	time
19870106	`findx2p`	1045	0.0168
	`gsolve`	1045	1650.7
19810601	`findx2p`	1118	0.0200
	`gsolve`	1118	1731.8
20100803	`findx2p`	1154	0.0185
	`gsolve`	1154	1804.6

(b) Random games with payoffs between -1 and 1

seed	function	check	time
19870106	`findx2p`	1045	0.0164
	`gsolve`	1045	1537.7
19810601	`findx2p`	1118	0.0212
	`gsolve`	1118	1744.8
20100803	`findx2p`	1154	0.0205
	`gsolve`	1154	1791.4

The results of the simulations can be found in table 3.1. Notice that for the same starting seed of the random generator both algorithms give the same value for check. Since theorem 2 states that in any random game the probability of finding a mixed equilibrium is $\frac{1}{9}$ it should make no difference if the games we examine are positive or have payoffs between -1 and 1. The table shows that it does not matter if the payoffs are only positive or between -1 and 1. The values for check are close to 1111. The simulations with the starting seed 19810601 come closest to the expected value we calculated for taking random numbers as payoff with a uniform distribution. For the starting seeds 19870106 and 20100803 we will calculate if the values found by the algorithm are acceptable using hypothesis testing.
We have the following situation:

- $n = 10000$ random games
- p = probability of "success" = p(a mixed equilibrium exist) = $\frac{1}{9}$
- X = the number of successes

We test $H_0 : X \sim B(n, p)$ with $n = 10000$ and $p = \frac{1}{9}$. Under H_0:

- $\mathrm{E}(X) = np = 1111\frac{1}{9} \approx 1111.11$

- $\text{Var}(X) = np(1-p) = 10000 \times \frac{1}{9}\frac{8}{9} = \frac{80000}{81} \approx 987.65$

Then $X \sim B(n,p) \approx N(\mu = np, \sigma = \sqrt{np(1-p)}) = N(1111.11, 31.43)$. The transformation

$$Z := \frac{X-\mu}{\sigma} = \frac{X - 1111\frac{1}{9}}{31.43} \sim N(0,1)$$

transforms all the observations of our random variable X into a new set of observations that are standard normally distributed. Now we are going to calculate the lower and upper bound for the 95% confidence interval. For the lower bound we get:

$$P(X \leq lb) = P(Z \leq \frac{lb - 1111\frac{1}{9}}{31.43}) = 0.025$$

meaning that

$$\frac{lb - 1111\frac{1}{9}}{31.43} = -1.96$$

so,

$$lb = 1049.5$$

and for the upperbound we get:

$$ub = 1172.7$$

thus the 95% confidence interval is

$$[1049.5; 1172.7]$$

The value 1045 lies outside the 95% confidence interval, so based on this value we would reject H_0. Therefore, we will do some more simulations to validate if 19870106 was a bad choice for a starting seed or that there is an error in our program. To validate, we do 100 simulations with the starting seeds 1 until 100. In figure 3.1 we plotted the results with the blue line showing the mean of the distribution and the black crosses representing the value for check for the corresponding seed. Notice, that depending on the starting seed the function may give a result close to mean value and the two outsider points are for the seeds 68 and 80 with values of 1220 and 1040 respectively. In appendix B.2, a table containing all values for check with the starting seed are given. The red and green line give the lower and upper bound values when taking a 95% confidence interval. We see that 5 points lie out of the range of the lower and upper bound and that is the expectation when taking a 95% confidence interval. So that indicates that our value for the seed 19870106 was correct and that there is no error in the program.
The big difference between the functions `findx2p` and `gsolve` is the time the algorithm needs to perform the calculations. The function `findx2p` is much faster, because the function only performs one calculation and does not use symbolic variables like `gsolve` does. From the simulations in this section, we can conclude that the function `gsolve` has a good performance since the results of the simulations for the functions `findx2p` and `gsolve` are the same.

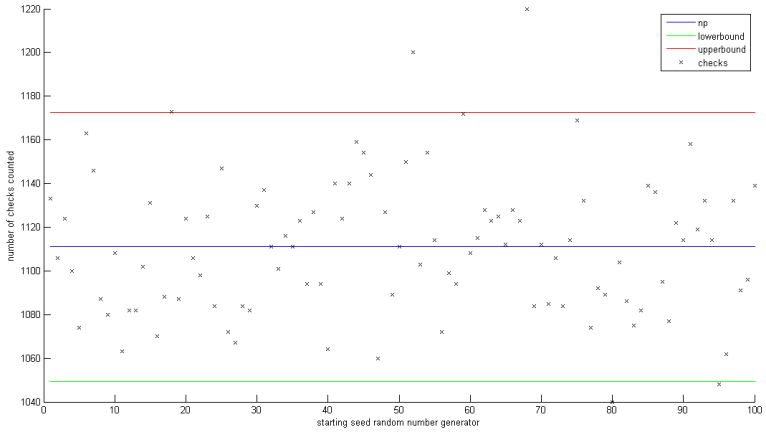


Figure 3.1: The number of mixed equilibria found in 10000 random games for 100 simulations

3.3 Determining the probability that a stationary equilibrium in a random 3-player game exists

The algorithm for finding the equilibria in 3-player games is a combination of the function `gsolve` and the algorithm for finding stationary equilibria in 2-player games if both players are indifferent. The main function is `threeplayers` given in appendix A.4, which has as an input a random 3-player quitting game and gives as an output all the equilibria for that game, if they exist.
The function starts with looking for pure stationary equilibria. A pure stationary equilibrium can exist in a 3-player quitting game. Look at the general 3-player quitting game with random payoffs given in formula (2.5) in section 2.2, a set of pure stationary strategies is an equilibrium if the payoffs meet the following requirements where c stands for continue and q stands for quit:

- for (c,c,c): $a_1 < 0$, $b_2 < 0$ and $c_4 < 0$,
- for (c,c,q): $a_4 > a_5$, $b_4 > b_6$ and $c_4 > 0$,
- for (c,q,c): $a_2 > a_3$, $b_2 > 0$ and $c_2 > c_6$,
- for (q,c,c): $a_1 > 0$, $b_1 > b_3$ and $c_1 > c_5$,
- for (c,q,q): $a_6 > a_7$, $b_6 > b_4$ and $c_6 > c_2$,
- for (q,c,q): $a_5 > a_4$, $b_5 > b_7$ and $c_5 > c_3$,
- for (q,q,c): $a_3 > a_2$, $b_3 > b_1$ and $c_3 > c_7$, and

- for (q,q,q): $a_7 > a_6$, $b_7 > b_5$ and $c_7 > c_3$.

For every set of pure stationary strategies, the function `checkpure` looks if the set of strategies is an equilibrium or not.
After looking for all pure stationary equilibria, the algorithm proceeds with looking for equilibria in which one of the players has a pure stationary strategy and the other players are indifferent. If the pure strategy of the player is playing continue all the time, then the game is reduced to a 2-player quitting game for the other players. The 2-player quitting game is then used as input for the function `findx2p` for finding the equilibrium for which the two players are indifferent. If the function `findx2p` has found an equilibrium, it still has to be checked if the equilibrium also is valid in the 3-player case, thus if the third player cannot do better by playing quit instead of continue for the actions of the other players.
If the pure stationary strategy of the player is play quit, then the 3-player quitting game can be reduced to a bimatrix game for the other two players. In that case, the function `findxbimatrix` tries to find an equilibrium for which the two players are indifferent. The bimatrix game used as an input for `findxbimatrix` for player 1 and 2 if player 3 is quitting, looks like:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix} \tag{3.2}$$

where the original bimatrix game looks like

	$1-y$	y
$1-x$	a_1, b_1	a_3, b_3
x	a_2, b_2	a_4, b_4

In the bimatrix game, player 1 is indifferent if

$$a_1(1-y) + a_3 y = a_2(1-y) + a_4 y$$

for y. This yields

$$y = \frac{a_1 - a_2}{a_1 - a_2 - a_3 + a_4}$$

For player 2 a similar formula can be found for x, namely

$$x = \frac{b_1 - b_3}{b_1 - b_2 - b_3 + b_4}$$

In the function `findxbimatrix` these formulas are generalized based on the matrix in formula (3.2) to the following for player i

$$x(i) = \frac{A(i,1) - A(i,i+1)}{A(i,1) - A(i,2) - A(i,3) + A(i,4)}$$

Like the case in which a player plays the pure stationary strategy continue, here also has to be checked if the pure stationary strategy quit for the third player

yields an equilibrium or if the third player is better off by deviating to continue. After looking for equilibria with all pure stationary strategies and the equilibria with one pure stationary strategy, the function `gsolve` is used to look for an equilibrium in which all players are indifferent.
For the testing we will use random games with positive payoffs, random games with payoffs between -1 and 1, and cyclicly symmetric random games. The function `randsym`, which can be found in appendix A.4, was written for creating these cyclicly symmetric random games.

3.3.1 The function `minmaxdif`

Since the function `gsolve` has long computation times because it uses symbolic variables, we have written the function `minmaxdif` to indicate if a game contains a completely mixed stationary equilibrium or not. The source code of the function `minmaxdif` can be found in appendix A.5.
The function `minmaxdif` has as an input a 3-player quittting game as given in equation (2.6) in section 2.2. The output for the function is a completely mixed equilibrium profile, called `x`, and the value `EPS`. `EPS` is the maximum a player can improve by deviation from `x`. So in fact an ϵ-equilibrium is calculated, where the value for ϵ is given in the variable `EPS`.
The function `minmaxdif` operates as follows: first the strategy set for each player is divided into 2m+1 gridpoints, with `m` being specified as input or using the default value `m`=5. The middlepoint of the complete grid for three players is $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Each gridpoint represents a strategy profile. After determining the gridpoints, the function `maxdif` (also found in appendix A.5) is called. The output of the function `maxdif` is the maximum a player can improve when deviating from the strategy profile. The function `minmaxdif` uses `maxdif` to calculate the maximum value for every gridpoint and takes the minimum of these values. The point which has the minimum value becomes the center of a finer grid, for which the same procedure is applied.
To validate the function `minmaxdif`, we will use the following examples:

Example 15:
Consider the random 3-player quitting game:

3

		2	
		1	
	1		-0.5298,-0.1615, 0.2557
		0.8597, -0.9646,0.1763	-0.7938, -0.8338,-0.8833

	1	-0.0124, 0.4061, 0.5199	0.3250,0.6236,0.9927
		0.6180,0.8210,-0.0250	0.3771,0.4556, -0.9655

When calling the function `minmaxdif` for this game, for `x` the following values are returned:

$$\mathbf{x} = \begin{bmatrix} 0.8890 & 0.8536 & 0.3166 \end{bmatrix}$$

Now we use the function `threeplayers` to check the result of `minmaxdif`. The

results returned by `threeplayers` are:

x^1	x^2	x^3
0.8889	0.8536	0.3167

So this game has only a completely mixed stationary equilibrium and as can be seen the values approximated by `minmaxdif` lie close to the values calculated by `gsolve`.

Example 16:
Consider another random 3-player quitting game:

3 = **1**

	2
	0.2534,0.8772, -0.1722
-0.9963, 0.8865, -0.6373	-0.1735, 0.3829,0.7865

3 = **2**

	2
-0.1199, -0.9902, 0.9728	-0.1947,0.8076,0.9008
0.9790,-0.5467,0.8822	-0.3723, -0.3240, -0.2538

When calling the function `minmaxdif` for this game, for `x` the following values are returned:

$$\mathbf{x} = \begin{bmatrix} 0.9814 & 0.6006 & 0.6543 \end{bmatrix}$$

Now we use the function `threeplayers` to check the result of `minmaxdif`. The equilibria returned by `threeplayers` are:

x^1	x^2	x^3
0	1	1
0.9814	0.6006	0.6543

So this game has not only a completely mixed stationary equilibrium for which the values approximated by `minmaxdif` are equal to the values calculated by `gsolve`, the game also has a pure stationary equilibrium which is (c,q,q).

Also an example of a cyclicly symmetric random game.

Example 17:
Consider the cyclicly symmetric random 3-player quitting game:

3 = **1**

	2
	-0.2576,0.1135,-0.5907
0.1135, -0.5907, -0.2576	-0.7123, -0.6242, -0.1957

3 = **2**

	2
-0.5907,-0.2576, 0.1135	-0.1957,-0.7123,-0.6242
-0.6242, -0.1957, -0.7123	-0.8947, -0.8947, -0.8947

When calling the function `minmaxdif` for this game, for `x` the following values

are returned:

$$\mathbf{x} = \begin{bmatrix} 0.3575 & 0.3575 & 0.3575 \end{bmatrix}$$

Now we use the function `threeplayers` to check the result of `minmaxdif`. The equilibria returned by `threeplayers` are:

x^1	x^2	x^3
0	0	1
0	1	0
1	0	0
0	0.4494	0.9546
0.9546	0	0.4494
0.4494	0.9546	0
0.3575	0.3575	0.3575

The function `gsolve` in `threeplayers` returns the same values as `minmaxdif`. The values for x^1, x^2 and x^3 are the same in the completely mixed equilibrium as we would expect for a cyclicly symmetric game. The function `threeplayers` also returns some other equilibria for this cyclicly symmetric random quitting game, and in these equilibria symmetry also shows.

As the above examples show, the functions `gsolve` and `minmaxdif` find the same strategy profiles `x` for the random games. The performance of the function `minmaxdif` is equal to the performance of `gsolve` and in section 3.2 we already concluded that `gsolve` finds the correct completely mixed equilibria.
Besides the quitting game, the function `minmaxdif` also has some optional inputs. The first optional input is `m`, used for dividing the grid into 2`m`+1 gridpoints. To test the influence of `m` on the simulations, we executed the following experiment: Always start a new simulation with the starting seed 19870601. Then for 20000 seconds let the program simulate cyclicly symmetric random games and count the number of times `minmaxdif` predicts the existance of a completely mixed equilibrium. For different values of `m` we get the following results:

`m`	# games simulated	# games with a completely mixed equilibrium
3	26865	15335
5	8488	4866
7	4242	2426
9	2144	1249
11	1309	756

If the value for `m` increases, the function `minmaxdif` can process less games. When we only look at the number of games `minmaxdif` can process in the given time, choosing `m`=3 would be logical. But for `m`¿3 the performance(# equilibria divided by # games) of finding completely mixed equilibria is higher. Looking at the performance and the number of games `minmaxdif` can process in the given time, choosing `m`=5 is a good default value.

Table 3.2: The number of games where the function `minmaxdif` finds a completely mixed equilibrium with `type` equals 0 and `type` equals 1. There are two tables, 3.2(a) for random games with payoffs between -1 and 1 and 3.2(b) for cyclicly symmetric random games.

(a) Random games with payoffs between -1 and 1

	# games	time (in seconds)
type = 0	1104	22801
type = 1	180	15201

(b) Symmetric random games

	# games	time (in seconds)
type = 0	5730	24705
type = 1	3877	16710

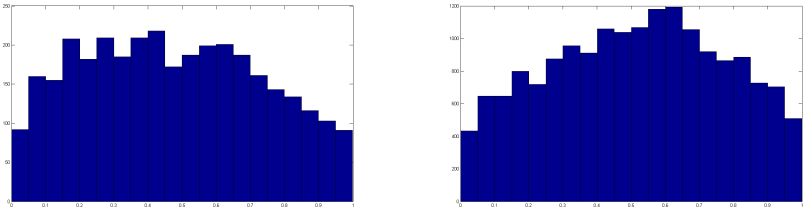
Another optional input parameter is `type`. The size of the finer grid is determined by `type`. The default value for `type` is 0. For `type`=0, the finer grid uses the complete grid from the neighbouring points of the new center. For `type`=1, the finer grid has a size of $\frac{1}{2}$ the length between the neighbouring points of the new center. The `type` parameter influences the number of games where `minmaxdif` finds a completely mixed equilibrium as table 3.2 shows.
The function `minmaxdif` works faster when `type`=1, but on the other hand for `type`=1 `minmaxdif` also misses a number of games with completely mixed equilibria. Since `type`=0 is more accurate and since the extra time consumption is reasonable, it is best to use `type`=0 for further experiments.
The last optional parameter is `a`. The parameter `a` is used to define when `minmaxdif` has to stop refining the grid. If the grid width is smaller then `a`, `minmaxdif` stops refining the grid and returns the values for `EPS` and `x`.

3.3.2 Stationary equilibria in 3-player quitting games

To see how the function `minmaxdif` works, we simulate 10000 random games with payoffs between -1 and 1 and also 10000 cyclicly symmetric random games. For each game, the function `minmaxdif` returns a value for `EPS` and `x`. We will use the values for `EPS` to determine a boundary for which we can say that a random game has a completely mixed equilibrium. The values for `EPS` can be stored in discrete intervals. Each discrete interval counts the number of games for which the value returned for `EPS` by `minmaxdif` returns lies in this interval. All intervals have a width of 0.000001. Table 3.3 the results of the simulations can be found.
As the table shows, the first interval contains a big number of games while the others are much lower. So taking 0.000001 as a boundary for determining the existence of a completely mixed equilibrium is a good choice. The results are as we would expect, since the value for `EPS` would be very small if the game

interval	random games with payoffs between -1 and 1	cyclicly symmetric random games
[0;0.000001)	1104	5730
[0.000001;0.000002)	38	2
[0.000002;0.000003)	22	2
[0.000003;0.000004)	13	1
[0.000004;0.000005)	7	1

Table 3.3: The results for the values of `EPS`<0.000005 when simulating 10000 games with the starting seed 19870601.



(a) Random games with payoffs between -1 and 1

(b) Cyclicly symmetric random games

Figure 3.2: The results for x values in the games where `EPS` was below 0.000001, when simulating 10000 random games with the starting seed 19870601

contains a completely mixed (ϵ-)equilibrium.
Besides the values for `EPS`, the function `minmaxdif` also returns an approximation of the strategy profile played in the completely mixed equilibrium. To get an idea about the distribution of the values of x, figure 3.2 shows the strategies for the games where `EPS`<0.000001.
In figure 3.2, a histogram shows the distribution of the values of x. Every x belongs to a specific bin in the histogram and no difference is made between the values for x for player 1,2 or 3. A bin is a discrete interval, where the height shows the frequency of observations in this interval. In figure 3.2 the bins have a width of 0.05, so each bin covers an interval of size 0.05. Both types of random games have a similar structure for x. The values close to 0 and 1 are less likely to occur, because for values close to 0 and 1 most likely the pure strategies continue and quit will contribute to an equilibrium.
For getting an idea about the percentage of random games with a completely mixed equilibrium we tested 30000 random games. For these 30000 games, we simulated three times 10000 games with different starting seeds. The exact results for each simulation can be found in appendix B.1. For the simulations we used the function `threeplayers` leaving out the part of gsolve for finding the equilibria where at least one player has a pure stationary strategy. After that we call `minmaxdif`, with the parameters `m`=5, `a`=$\frac{1}{11000000}$ and `type`=0, to see if

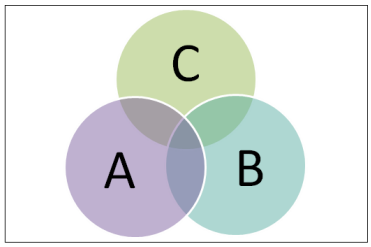


Figure 3.3: The categories in which the found equilibria can be divided: A: games with pure equilibria, B: games with equilibria where two players randomize, and C: games with completely mixed equilibria.

	category I	category II	category III
$A \cup B \cup C$	27246	24777	29532
C	3237	3158	17026
$C \backslash (A \cap B)$	426	772	3222
$(C \cap A) \backslash B$	2249	1841	12506
$(C \cap B) \backslash A$	125	150	5
$C \cap A \cap B$	437	395	1293
$(A \cup B) \backslash C$	24009	21619	12506
$A \backslash (B \cap C)$	20748	18341	12429
$B \backslash (A \cap C)$	755	1000	2
$(A \cap B) \backslash C$	2506	2278	75

Table 3.4: Equilibria found when simulating 30000 random games divided into the categories as shown by figure 3.3. The games can also be divided into three different categories: I: positive random games, II: random games with payoffs between -1 and 1, and III: cyclicly symmetric random games

the game contains a completely mixed equilibrium. We chose the value `EPS`, in such a way that the grid is refined ten times. The results of the simulations can be found in table 3.4.
For cyclicly symmetric random games, approximately 98% of the games have a stationary equilibrium and in 57% of the games a completely mixed equilibrium exists. If a cyclicly symmetric random game has a completely mixed equilibrium, the chance of having also a pure equilibrium is very high. On the other hand, if a cyclicly symmetric random game has a completely mixed equilibrium, it is very rare that the game also has an equilibrium where two players randomize and the third plays a pure stationary strategy. If a cyclicly symmetric random games does not have a completely mixed equilibrium, it most likely has a pure equilibrium while equilibria where two players randomize are less common in cyclicly symmetric random games.
For random games with positive payoffs and payoffs between -1 and 1, the probability of having a stationary equilibrium is lower than for cyclicly symmetric

random games. Approximately 91% of the positive random games have a stationary equilibrium and for the random games with payoffs between -1 and 1 this is approximately 83%. The percentage of games with a completely mixed equilibrium is in both categories approximately 11%. There are some differences: in positive random games there is a bigger chance of having also a pure equilibrium if the game has a completely mixed equilibrium and for random games with payoffs between -1 and 1, the probability that the completely mixed equilibrium is the only equilibrium in the game is bigger.

3.4 Some examples of 3-player quitting games

In this section we discuss some specific examples of 3-player quitting games.

Example 18:
Consider the symmetric 3-player quitting game:

3

		2		**2**
1		0,0,0	0,0,0	1,1,1
	0,0,0	1,1,1	1,1,1	0,0,0

In this game the following equilibria exist:

x^1	x^2	x^3
0	0	0
0	1	1
1	0	1
1	1	0
1	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	1	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	1
$\frac{3}{2} - \frac{\sqrt{3}}{2}$	$\frac{3}{2} - \frac{\sqrt{3}}{2}$	$\frac{3}{2} - \frac{\sqrt{3}}{2}$

The game has 8 stationary equilibria.
(c,c,c) is an equilibrium, because by deviating the players also get payoff 0 so deviating is not profitable for them.(c,q,q), (q,c,q) and (q,q,c) are all equilibria, because of the symmetric of the game and in these matrix entries the players have payoff 1 while by deviation they receive payoff 0. $(c, \frac{1}{2}, \frac{1}{2})$, $(\frac{1}{2}, c, \frac{1}{2})$ and $(\frac{1}{2}, \frac{1}{2}, c)$ are all equilibria because $(\frac{1}{2}, \frac{1}{2})$ is an equilibrium for the 2-player quitting game and the third player cannot gain anything by deviating to quit.
A completely mixed equilibrium, like $(\frac{3}{2} - \frac{\sqrt{3}}{2}, \frac{3}{2} - \frac{\sqrt{3}}{2}, \frac{3}{2} - \frac{\sqrt{3}}{2})$ only exists if all players are indifferent. For all players to be indifferent in a symmetric game, $x^1 = x^2 = x^3$. So we can calculate the completely mixed equilibrium by assuming that player 1 and 2 both quit with probability x. Player 3 is indifferent for a value of x where his payoff for playing continue equals the payoff for playing

quit. Now, continue yields player 3 a payoff of:

$$\text{continue:}\frac{x^2}{1-(1-x)^2}$$

and quit yields a payoff of:

$$\text{quit:}2x(1-x)$$

Solving the equation continue = quit, yields $x = \frac{3}{2} - \frac{\sqrt{3}}{2}$ which are the values for x^1, x^2, x^3 in the completely mixed equilibrium.

The game in the following example is similar to the game in example 3 in chapter 1. The only difference is that the payoffs with value 3 are changed to payoffs with value 2.

Example 19:
Consider the cyclicly symmetric 3-player quitting game:

3

		2		**2**
1		0,1,2	2,0,1	1,1,0
	1,2,0	1,0,1	0,1,1	0,0,0

The stationary equilibrium for this game is:

x^1	x^2	x^3
ϵ	ϵ	ϵ

When trying to find the equilibrium in this game, the function `threeplayers` has no solutions. Therefore, we used the function `minmaxdif` to determine if a completely mixed equilibrium exists. When solving `[EPS,x]=minmaxdif(A, 5, 1/11000000, 0)`, where `A` is the game matrix, we find the following:

$$\texttt{a} = 1.4964 \cdot 10^{-8}$$

and

$$\texttt{x} = 10^{-8} \cdot \begin{bmatrix} 0.8269 & 0.8269 & 0.8269 \end{bmatrix}$$

Because the values for `x` are smaller than the value for `EPS` and close to zero, we may assume that the equilibrium is quitting with probability ϵ for all players. When players quit with probability ϵ, the probability that two players quit at the same time is negligible. If the probability with which the players quit increases, then the chance of two players quitting at the same time also increases. Therefore, quitting with probability ϵ is an equilibrium. Looking at the game it shows that quitting with a small probability is better for all players than playing continue all the time.

Chapter 4

Conclusion and recommendations

Before giving an overall conclusion about the problem statement, we will first answer reseach questions.
The first research question is: Is it possible to set up an algorithm for finding stationary ($\epsilon-$)equilibria in 2-player quitting games? Since we wrote an algorithm which finds all stationary (ϵ-)equilibria in a 2-player game, it is possible to set up an algorithm for finding these equilibria. The simplest way to do so, is to use calculate for every stationary strategy of one player, the other player's stationary best replies. We tested the algorithm with different games, like games with only positive or negative payoffs, symmetric games and games with payoffs between -1 and 1. In every situation our algorithm `twoplayers` returned the correct equilibria, which we verified by doing the calculations manually for finding the equilibrium.
The second research question is: Is it possible to set up an algorithm that correctly identifies a stationary equilibrium in a 3-player quitting game, or that says that there exists no stationary equilibrium? For 3-player quitting games we have written an algorithm that can identify a stationary equilibrium if it exists. Here we also used different types of games to test the working, like symmetric random games and random games with payoffs between -1 and 1. The algorithm `threeplayers` returns the equilibria for which one or more players have a pure stationary strategy and also the equilibria for which all players randomize. So it can correctly identify all stationary equilibria in a 3-player quitting game.
The third research question is: Can an algorithm prove the existence of equilibria in 3-player quitting games? In specific 3-player quitting games our algorithms are able to find the stationary equilibria, but this does not apply to all 3-person quitting games. So the algorithms we wrote are not able to prove the existence of equilibria in general 3-person quitting games but only in specific cases.
Our problem statement is : 'Using Matlab to find stationary (ϵ-)equilibria in 2- and 3-player quitting games'. Matlab is a useful tool when writing algorithms

to find stationary (ϵ-)equilibria in quitting games. Matlab is able to identify and find all possible stationary equilibria that exist in 2- and 3-player quitting games. For finding completely mixed stationary equilibria in 3-player quitting games the only drawback of using Matlab is the time it takes before Matlab has calculated such an equilibrium.
The research, conducted in this thesis, has led to some interesting results in 2- and 3-player quitting games. These games are the simplest kind of quitting games, because the verification of the results can be done by hand. For quitting games with $n \geq 4$ players, the games get more complicated. It might be interesting to see if our algorithms can be extended in such a way, that they are also applicable for larger and more complex quitting games. For finding the completely mixed equilibria in quitting games with $n \geq 4$ players, this might not be a problem since `gsolve` can be used. For finding the stationary equilibria where at least one player does not randomize over the action the situation is different and it might be a challenge to change our algorithms such that they are suitable to use.
Since we only focussed on finding the stationary equilibria in quitting games, another interesting research area would be the finding of cyclic equilibria in n-player quitting games. From examples 3 and 4 in chapter 1, we know that cyclic equilibria can exist in quitting games that have no stationary equilibrium. Since we can set up algorithms for finding stationary equilibria, a challenge would be to set up an algorithm that detects a cyclic equilibrium.

Bibliography

[1] D. Blackwell and T.S. Ferguson. The big match. *The Annals of Mathematical Statistics*, 39(1):159–163, 1968.

[2] H. Everett. Recursive games. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games, Vol. III*, Annals of Mathematics Studies 39, pages 47–78. Princeton University Press, 1957.

[3] J. Flesch, J. Kuipers, G. Schoenmakers, and K. Vrieze. Subgame-perfection in free transition games. Research memorandum, Maastricht University, 2011.

[4] J. Flesch, F. Thuijsman, and K. Vrieze. Cyclic markov equilibria in stochastic games. *International Journal of Game Theory*, 26:303–314, 1997.

[5] J. Flesch, F. Thuijsman, and O.J. Vrieze. Recursive repeated games with absorbing states. *Mathematics of Operations Research*, 21(4):1016–1022, November 1996.

[6] D. Gillette. Stochastic games with zero stop probabilities. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games, Vol. III*, Annals of Mathematics Studies 39, pages 197–208. Princeton University Press, 1957.

[7] J. Kuipers, J. Flesch, G. Schoenmakers, and K. Vrieze. Pure subgame-perfect equilibria in free transition games. *European Journal of Operational Research*, 199(2):442–447, December 2009.

[8] E. Solan. Three-player absorbing games. *Mathematics of Operations Research*, 24(3):669–698, August 1999.

[9] E. Solan and N. Vieille. Quitting games. *Mathematics of Operations Research*, 26(2):265–285, May 2001.

[10] E. Solan and N. Vieille. Quitting games - an example. *International Journal of Game Theory*, 31:365–381, 2002.

[11] O.J. Vrieze and F. Thuijsman. On equilibria in repeated games with absorbing states. *International Journal of Game Theory*, 18:293–310, 1989.

Appendix A

Source code of the algorithms

Here, the Matlab code for the algorithms is given. First some functions for the algorithm which finds the equilibria for two players, then the algorithm for finding the equilibria in games with 3 players. After that some functions used for the validation of the algorithm for 3 players.

A.1 Finding equilibria for two players

The algorithm for two players consist of a main function where the game is inputted, this function then calls the other functions that are needed to calculate the equilibria in the 2-player game. Below the main function called `twoplayers` is given.

Function 1 twoplayers

Input: game
Output: equilibrium

```
function eq=twoplayers(game)
% 2 players
% look at the best reply from one player to a stationary strategy
% of the other and see if there are matches between them
% the match is the epsilon-equilibrium of the game
% change the game in two subgames, one for each player, which
% later can be used to calculate the best reply
% player1's game
game1 = [game(1,1), game(1,3); game(2,1), game(2,3)]';
% player2's game
game2 = [game(1,2), game(1,4); game(2,2), game(2,4)];
```

```
% initialize the matrix for storing a strategy with the best
% reply, the left column is action player 1, the right column
% is action player 2, this is useful for later comparison in
% finding equilibrium
s1 = []; % strategies for player 1, reply player 2
s2 = []; % strategies for player 2, reply player 1
% there are three possible actions to be played, namely continue,
% quit and a mixed action of continue and quit
for i=1:3
   % first the reply for the stationary strategy
   action = i;
   [action1, reply2] = findreply2p(action, game2);
   [action2, reply1] = findreply2p(action, game1);
   % store results in strategies matrices
   s1 = [s1; action1, reply2];
   s2 = [s2; reply1, action2];
end
% find equilibrium
eq=findeq2p(s1,s2);
```

The first and most important part of finding the equilibria in a game for two players, is to look at one player's stationary strategy and find the best reply for the other player. The function described below, tries to find the best reply for a player.

Function 2 findreply2p

Input: action, game
Output: action, reply

```
function [action, reply] = findreply2p(action, game)
eps=10^-4;
if(action == 1) % continue
   action = [0,0];
   if(game(1,1)== game(1,2))
      reply = [0,1];
   elseif(game(1,1) > game(1,2))
      reply = [0,0];
   else
      reply = [0+eps,1];
   end
elseif(action == 2) %quit
   action = [1,1];
   if(game(2,1)==game(2,2))
      reply = [0,1];
   elseif(game(2,1)>game(2,2))
```

```
      reply = [0,0];
   else
      reply = [1,1];
   end
else % mixed action of continue and quit
% if the payoff for continue is higher or equal, then the sum of
% the payoffs for quit, the player should always continue
% if the payoffs for quit are positive and always higher then the
% payoff for continue, the player should quit all the time
% if the above situations don't occur, the player should do a
% mixed action as best reply
   b = game(1,2);
   c = game(2,1);
   d = game(2,2);
   if(b==d==c)
      action=[eps,1-eps];
      reply=[0,1];
   elseif(c>= max(b,d))
      action = [eps,1-eps];
      reply = [0,0];
   elseif(c<= min(b,d))
      action = [eps,1-eps];
      reply = [1,1];
   else
   % solve the equation:  c = x*d + (1-x)*b (x is act)
      act = (b-c)/(b-d);
      % if act is between 0 and 1, then for the value of act, the
      % reply can be anything, and look also for a small added or
      % subtracted value from act what the best reply is else
      % there is only one best reply for the action [x, 1-x]
      action = [act, act; eps, act-eps;act+eps, 1-eps];
      r1 = [0,1];
      if(b>d)
         r2=[1, 1; 0, 0];
      else
         r2=[0, 0; 1, 1];
      end
      reply = [r1;r2];
   end
end
```

After searching for the best replies, there are two matrices created which contain the strategy pairs for both players. For an equilibrium, look for the strategy pairs that are best replies to each other or in other words, look for strategy pairs that both players have in common. For that the following function is used:

Function 3 findeq2p
Input: 2 lists of strategies
Output: equilibrium

```
function finaleq = findeq2p(s1, s2)
eps=10^-4;
l1 = size(s1,1);
l2 = size(s2,1);
eq=[];
% compare the two matrices with strategies to see, where the
% equilibrium is
for j=1:l1
   for k=1:l2
   % do comparing
      overlap1=intervaloverlap(s1(j,1:2),s2(k,1:2));
      overlap2=intervaloverlap(s1(j,3:4),s2(k,3:4));
      if(( isempty(overlap1))&&( isempty(overlap2)))
         eq=[eq; overlap1, overlap2];
      end
   end
end
m=size(eq,1);
% if the size of eq is bigger then 1, look if there as an action
% for which the other player has multiple best replies and
% combine these replies (if possible) into 1 reply
if(m>1)
% first check if player 1 strategies can be combined
   eqtemp=sortrows(eq, [3 1]);
   eq1=findadjacent(eqtemp,eps,1);
   % check if strategies of player 2 can be combined
   eqtemp2=sortrows(eq1, [1 3]);
   eq2=findadjacent(eqtemp2, eps,2);
   eq=eq2;
end
finaleq=eq;
```

The previous function uses two other functions, namely `intervaloverlap` and `findadjacent`. The function `intervaloverlap` checks if the intervals have some overlapping values and the overlap is then part of the equilibria.

Function 4 intervaloverlap
Input: Two pairs of intervals
Output: The overlap in intervals

```
function overlap = intervaloverlap(int1, int2)
x1=int1(1,1);
x2=int2(1,1);
y1=int1(1,2);
y2=int2(1,2);
overlap=[];
if(x1>=x2)
   if(y2>=y1)
      overlap=int1;
   elseif((y2<=y1)&&(x1<=y2))
      overlap=[x1,y2];
   end
elseif(x1<=x2)
   if((y2>=y1)&&(x2<=y1))
      overlap=[x2,y1];
   elseif(y2<=y1)
      overlap=int2;
   end
end
```

The function `findadjacent` looks if the intervals ensuring the equilibria can be combined into a bigger interval for the equilibrium.

Function 5 findadjacent

Input: The list with equilibria, the value for epsilon and the number of players
Output: The list with equilibria

```
function eqout = findadjacent(eqin, eps, playerno)
% this function checks if for one actionpair, the other players
% actions can be grouped together
m=size(eqin,1);
counter=1;
% ensures that there are no more loops than size of matrix
i=1; % keeps track of index in matrix
while(counter<m)
   if(playerno==1)
      eq1=eqin(i,3:end);
      eq2=eqin(i+1,3:end);
   elseif(size(eqin,2)==(2*playerno))
      eq1=eqin(i,1:(end-2));
      eq2=eqin(i+1,1:(end-2));
   else
   eq1=[eqin(i,1:(2*(playerno-1))),eqin(i,(2*playerno+1):end)];
   eq2=[eqin(i+1,1:(2*(playerno-1))),eqin(i,(2*playerno+1):end)];
   end
```

```
   str1=num2str(eq1,4); % values are strings with 4 digits
   str2=num2str(eq2,4);
   % check if the strategies for the first player(s) in row i
   % and row i+1 are the same before checking if the strategy
   % of the last player in row i and row i+1 is a follow up
   % (difference smaller or equal to eps)
   if(strcmp(str1,str2))
      if((eqin(i+1,(2*playerno-1))-eqin(i,(2*playerno)))<=eps)
         eqin(i,(2*playerno))=eqin(i+1,(2*playerno));
         eqin(i+1,:)=[];
      else
         i=i+1;
      end
   else
      i=i+1;
   end
   counter=counter+1;
end
eqout=eqin;
```

A.2 Finding equilibria for 3 players

The algorithm for finding equilibria in games with 3 players consists of two functions. The main function is `gsolve`, where the game is inputted and the sets of players that continue or quit.

Function 6 gsolve

Input: game, set of players that continue, set of players that quit
Output: values for x for indifferent players

```
function [Q,B]=gsolve(A,Sc,Sq)
d=size(A); n=d(1); N=d(2);
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 positive
x=[x1 x2 x3 x4 x5 x6 x7 x8 x9];
x=x(1:n);
c=1; I=[];
for i=1:n
   if bitget(Sq,i)
      x(i)=1;
   elseif bitget(Sc,i)
      x(i)=0;
   else
      I=[I,i];
```

```
   end
   c=c*(1-x(i));
end
B=[];
for i=I
   y1=x; y1(i)=1; y1=mgame(y1);
   y0=x; y0(i)=0; y0=mgame(y0);
   B=[B;(1-c/(1-x(i)))*A(i,:)*y1'-A(i,:)*y0'];
end
Q=solve(B);
```

The function `gsolve` uses the function `mgame` where the vector with probabilities is calculated according to the values for $x(i)$ for player i which is indifferent.

Function 7 mgame

Input: vector with probabilities
Output: vector with probabilities

```
function y=mgame(q,i)
if nargin>1
   if i<0
      q(-i)=eps;
   else
      q(i)=1;
   end
end
y=q(1); c=1;
for i=2:length(q)
   c=c*(1-q(i-1));
   y=[y*(1-q(i)), q(i)*c, y*q(i)];
end
```

A.3 Functions for the validation of `gsolve`

For the validation of the function `gsolve`, the following function was written which uses as an input the game in the same form as the function `gsolve` and as an output the values for x, so the probabilities with which players quit, are given:

Function 8 findx2p
Input: game
Output: probabilities

```
function Q = findx2p(A)
Q = zeros(1,2);
for i=1:2
   Q(i)=(A(i,i)-A(i,3-i))/(A(i,i)-A(i,3));
end
```

A.4 Algorithm for finding equilibria in 3-player games

The main function for this algorithm is `threeplayers`. The function uses as an input a game as given in equation 2.5 and the output is a matrix with equilibria if they exist.

Function 9 threeplayers
Input: game
Output: equilibrium

```
function eq = threeplayers(A)
eq=[];
% first check if the game has a pure stationary equilibrium
for i=0:1 % action for player 1
   for j=0:1 % action for player 2
      for k=0:1 % action for player 3
         x=[i, j, k];
         tempeq = checkpure(A, x);
         eq=[eq;tempeq];
      end
   end
end
% check if the game has an equilibrium for which two players are
% indifferent and the third player plays a fixed action
for i=1:3
   Atempq=[];
   Atempc=[];
   for j=1:7
      if bitget(j,i)
         Atempq=[Atempq, A(:,j)];
```

```
      else
         Atempc=[Atempc, A(:,j)];
      end
   end
   Atempq(i,:)=[];
   Atempc(i,:)=[];
   Qc=findx2p(Atempc);
   Qq=findxbimatrix(Atempq);
   xc = Qc(1);
   yc = Qc(2);
   if(xc>0 && xc<1 && yc>0 && yc<1)
      tempeq = checkcontinue(A, Qc, i);
      eq=[eq; tempeq];
   end
   xq = Qq(1);
   yq = Qq(2);
   if(xq>0 && xq<1 && yq>0 && yq<1)
      tempeq = checkquit(A, Qq, i);
      eq=[eq; tempeq];
   end
end
% check if the game has an equilibrium for which all players are
% indifferent Q = gsolve(A,0,0);
if  isempty(Q)
   x1=double(Q.x1);
   x2=double(Q.x2);
   x3=double(Q.x3);
   if(x1<1 && x2<1 && x3<1)
      eq=[eq;x1 x2 x3];
   end
end
```

For finding strategy sets if one of the players has a fixed strategy, the functions `findx2p` and `findxbimatrix` are used. The function `findx2p` is the function described in section A.3. The function `findxbimatrix` is similar, but now the input is a 2-player bimatrix game.

Function 10 findxbimatrix
Input: game
Output: probabilities

```
function Q = findxbimatrix(A)
Q = zeros(1,2);
for i=1:2
   Q(i)=(A(i,1)-A(i,i+1))/(A(i,1)-A(i,2)-A(i,3)+A(i,4));
```

```
end
```

The main function `threeplayers` also uses the following three functions to check if the set of strategies found is indeed an equilibrium of the 3-player game provided as input to the function. The first function is `checkpure` which looks if a set of pure strategies are an equilibrium, if not an empty matrix is given as output.

Function 11 checkpure

Input: game, set of strategies
Output: equilibrium

```
function eq = checkpure(A, x)
eq=[];
bool = 1;
for i=1:3
   x2=x;
   x2(i)=mod(x(i)+1,2);
   if(x== zeros(1,3))
      a1 = 0;
   else
      index = x(1)+x(2)*2+x(3)*4;
      a1 = A(i, index);
   end
   if(x2 == zeros(1,3))
      a2=0;
   else
      index = x2(1)+x2(2)*2+x2(3)*4;
      a2=A(i, index);
   end
   if  (a1 >= a2)
      bool = 0;
      break;
   end
end
if bool
   eq=x;
end
```

The functions `checkcontinue` and `checkquit` are used to check if the set of strategies found when one of the players plays the pure strategy continue or quit is an equilibrium.

```
function eq = checkcontinue(A, Q, i)
eq=[];
```

Function 12 checkcontinue
Input: game, set of strategies, playernumber
Output: equilibrium

```
xc = zeros(1,3);
xq = zeros(1,3);
xc(i) = 0;
xq(i) = 1;
c= 1- (1-Q(1))*(1-Q(2));
counter = 1;
for j=1:3
   if  (i==j)
      xc(j)=Q(counter);
      xq(j)=Q(counter);
      counter = counter+1;
   end
end
xc2=mgame(xc);
xq2=mgame(xq);
con = A*xc2'/c;
quit = A*xq2';
if con>=quit
   eq=xc;
end
```

The function `checkquit` is similar to the function `checkcontinue` described above, the only difference is that now you look if `con<=quit`.
Since we not only simulate random games, but also symmetric random games, we have written a short function `randsym` for creating a symmetric random game, which looks as follows:

	a_5, a_1, a_4
a_1, a_4, a_5	a_6, a_7, a_2

a_4, a_5, a_1	a_2, a_6, a_7
a_7, a_2, a_6	a_3, a_3, a_3

or as input for the algorithms,

the game is returned by `randsym` in the form as given in equation 2.5:

$$\begin{bmatrix} a_1 & a_5 & a_6 & a_4 & a_7 & a_2 & a_3 \\ a_4 & a_1 & a_7 & a_5 & a_2 & a_6 & a_3 \\ a_5 & a_4 & a_2 & a_1 & a_6 & a_7 & a_3 \end{bmatrix}$$

Function 13 randsym
Output: symmetric random game

```
function G=randsym()
```

```
% computes a random symmetric 3-person quitting game
a=rand(1,7)*2-1;
G=[a([1 5 6 4 7 2 3]); a([4 1 7 5 2 6 3]); a([5 4 2 1 6 7 3])];
```

A.5 Algorithm for finding mixed equilibria in 3-player games

Since the function `gsolve` tries to find the exact completely mixed equilibria, we have written a different function called `minmaxdif` which estimates if a random game has a completely mixed stationary equilibrium without calculating it exactly.

Function 14 minmaxdif
Input: game, indicator for number of grid points, value for stopping, type
Output: difference, set of stationary strategies

```
function [EPS,x,n]=minmaxdif(A,m,a,type)
% input A: a 3-person quitting game.
% output 'EPS','x':  in stationary strategy 'x' the two payoffs
% p0 and p1 for any player after deviating to continue, resp.
% quit, differ by at most 'EPS', and there is no other strategy
% with smaller 'EPS'.
if nargin<4, type=0; end
if nargin<3, a=0.0000001; end
if nargin<2, m=5; % with m=5, meaning 11 gridpoints van -5 t/m 5
X=[];
for i=-m:m, for j=-m:m, for k=-m:m, X=[X;[i j k]]; end, end, end
if type
   m=1/(2*m+1); % grid shrinks with m at each step
   X=X*m; % regular grid of overall width 1 minus the border
else
   m=1/(m+1); % grid shrinks with m at each step
   X=(X*m)/2;
end
w=1; % grid width of the search space
x=[1 1 1]/2; % starting point
n=1;
while w>a
   n=n+1;
   X1=[w*X(:,1)+x(1),w*X(:,2)+x(2),w*X(:,3)+x(3)];
   SS=maxdif(A,X1);
   [EPS,i]=min(SS);
   x=X1(i,:);
```

```
   w=w*m;
end
```

The function `minmaxdif` uses the function `maxdif` to calculate the value by which the players can improve when deviating from the given strategies.

Function 15 maxdif
Input: game, sequence of stationary strategies, eps
Output: the maximum the players can improve by deviating

```
function SS=maxdif(A,X)
% Input:
% A is an n-person quitting game, and
% X a sequence of stationary strategies
% Considering for each player the payoffs p0 and p1, the
% first when deviating by playing continue and the second by
% playing quit, the outcome SS(j) stores the maximum |p0-p1|
% over all players.
TOLR=10*eps;
d=size(A); n=d(1); N=d(2);
d=size(X); m=d(1);
SS=zeros(1,m);
for j=1:m
   x=X(j,:);
   c=(1-x(1)); for i=2:n, c=c*(1-x(i)); end
   S=0;
   for i=1:n
      y1=x; y1(i)=1; y1=mgame(y1); p1=A(i,:)*y1';
      if x(i)>1-TOLR, ci=0; else ci=c/(1-x(i)); end
      y0=x; y0(i)=0; y0=mgame(y0); p0=(A(i,:)*y0')/(1-ci);
      S=max([S,p1-p0,p0-p1]);
   end
   SS(j)=S;
end
```

Appendix B

Tables with results of simulations

For reproducability of the simulations for the experiments, some tables with results are given here. First a table with results for determining the probability that a stationary equilibrium exist in a random 3-player game is given and after that a table with results for the validation of `gsolve`.

B.1 Results for stationary equilibria in random 3-player games

The results for finding stationary equilibria in random 3-player quitting games, for each simulation of 10000 games with a specified starting seed.

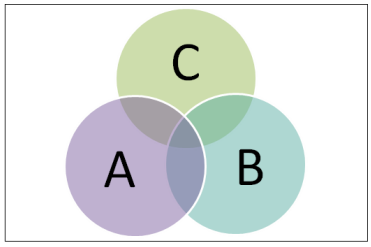


Figure B.1: The categories in which the found equilibria can be divided: A: games with pure equilibria, B: games with equilibria where two players randomize, and C: games with completely mixed equilibria.

seed	19870106	19810601	20100803
$A \cup B \cup C$	9053	9118	9075
C	1077	1085	1075
$C \backslash (A \cap B)$	149	136	141
$(C \cap A) \backslash B$	743	764	742
$(C \cap B) \backslash A$	37	49	39
$C \cap A \cap B$	148	136	153
$(A \cup B) \backslash C$	7976	8033	8000
$A \backslash (B \cap C)$	6917	6914	6917
$B \backslash (A \cap C)$	244	254	257
$(A \cap B) \backslash C$	815	865	826

Table B.1: The number of equilibria found when simulating 10000 positive random games, divided into the categories as shown by figure B.1.

seed	19870106	19810601	20100803
$A \cup B \cup C$	8235	8251	8291
C	1047	1056	1055
$C \backslash (A \cap B)$	276	235	261
$(C \cap A) \backslash B$	591	643	607
$(C \cap B) \backslash A$	46	57	47
$C \cap A \cap B$	134	121	140
$(A \cup B) \backslash C$	7188	7195	7236
$A \backslash (B \cap C)$	6124	6069	6148
$B \backslash (A \cap C)$	330	335	335
$(A \cap B) \backslash C$	734	791	753

Table B.2: The number of equilibria found when simulating 10000 random games with payoffs between -1 and 1, divided into the categories as shown by figure B.1.

seed	19870106	19810601	20100803
$A \cup B \cup C$	9850	9850	9832
C	5606	5811	5609
$C \backslash (A \cap B)$	1075	1081	1066
$(C \cap A) \backslash B$	4138	4269	4099
$(C \cap B) \backslash A$	0	1	4
$C \cap A \cap B$	393	460	440
$(A \cup B) \backslash C$	4244	4039	4223
$A \backslash (B \cap C)$	4220	4018	4191
$B \backslash (A \cap C)$	0	0	2
$(A \cap B) \backslash C$	24	21	30

Table B.3: The number of equilibria found when simulating 10000 cyclicly symmetric random games, divided into the categories as shown by figure B.1.

B.2 Results for the validation of `gsolve`

Here is a list with the values as shown in figure 3.1 in section 3.2:

seed	check	seed	check	seed	check	seed	check
1	1133	2	1106	3	1124	4	1100
5	1074	6	1163	7	1146	8	1087
9	1080	10	1108	11	1063	12	1082
13	1082	14	1102	15	1131	16	1070
17	1088	18	1173	19	1087	20	1124
21	1106	22	1098	23	1125	24	1084
25	1147	26	1072	27	1067	28	1084
29	1082	30	1130	31	1137	32	1111
33	1101	34	1116	35	1111	36	1123
37	1094	38	1127	39	1094	40	1064
41	1140	42	1124	43	1140	44	1159
45	1154	46	1144	47	1060	48	1127
49	1089	50	1111	51	1150	52	1200
53	1103	54	1154	55	1114	56	1072
57	1099	58	1094	59	1172	60	1108
61	1115	62	1128	63	1123	64	1125
65	1112	66	1128	67	1123	68	1220
69	1084	70	1112	71	1085	72	1106
73	1084	74	1114	75	1169	76	1132
77	1074	78	1092	79	1089	80	1040
81	1104	82	1086	83	1075	84	1082
85	1139	86	1136	87	1095	88	1077
89	1122	90	1114	91	1158	92	1119
93	1132	94	1114	95	1048	96	1062
97	1132	98	1091	99	1096	100	1139