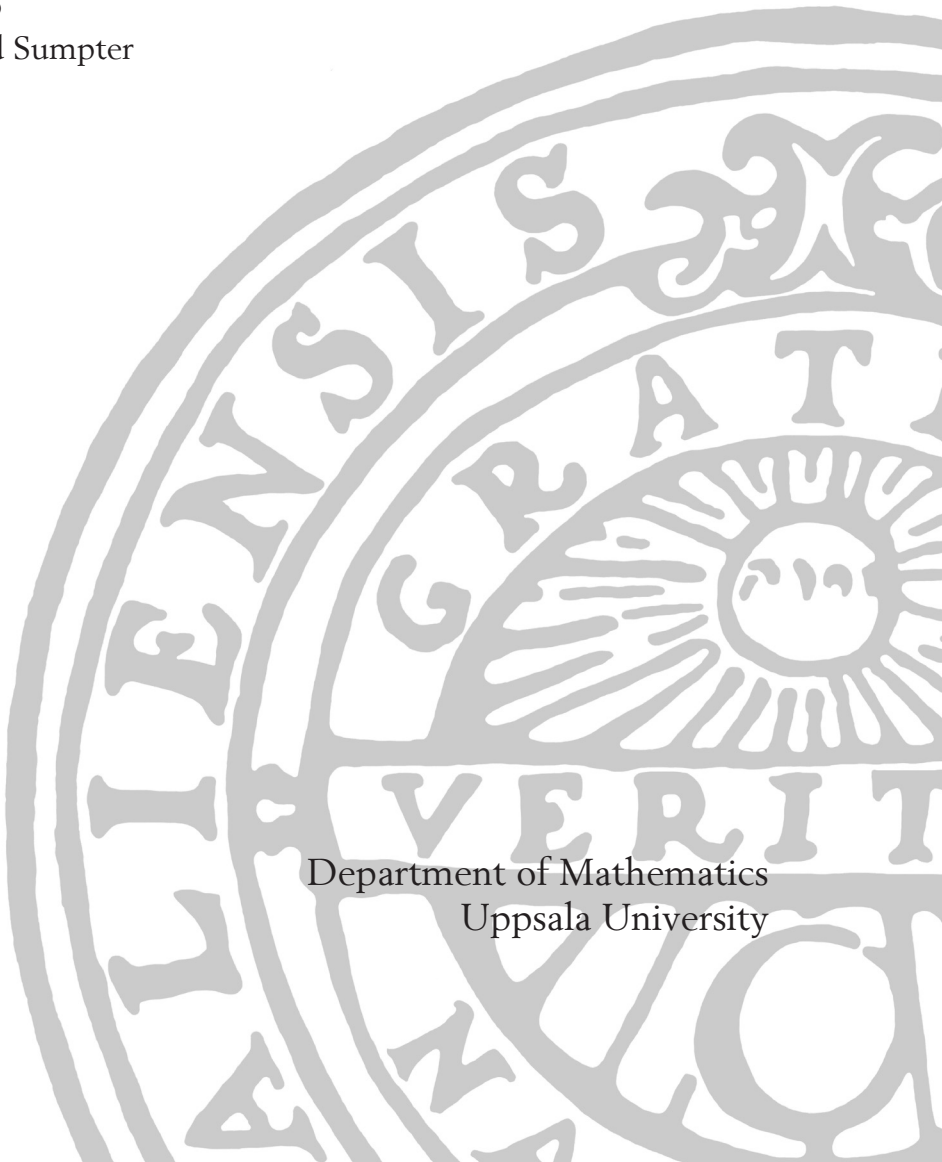
UPPSALA
UNIVERSITET

U.U.D.M. Project Report 2014:6

Analysis of a bluffing game

Adam Malik

Examensarbete i matematik, 15 hp
Handledare och examinator: David Sumpter
Februari 2014



Department of Mathematics
Uppsala University

Analysis of a bluffing game

Adam Malik

February 15, 2014

Contents

Abstract

The goal of my bachelors thesis was to define and analyze a simplified version of poker which does not have an optimal strategy, using evolutionary game theory. I used both a deterministic model containing the replicator equations and a stochastic model. The game Adam poker does not have an optimal strategy. For every strategy there is at least one counterstrategy. During the analysis I found out which strategies are the best ones in a mixed population, even though they are not optimal.

1 Introduction

1.1 Poker terminology

Here are brief descriptions of the poker terms I will use.

Expected value is the expected win or loss of some action or strategy in poker. It also has the same meaning as expected value in probability theory.

Raise/bet is when a player puts money into the pot. In normal poker this requires the other players to put in as much or more. In Adam poker this is not the case.

Check is when a player does not raise.

Call is when a player calls a bet. To do this the player needs to put in the same amount of money as the player making the bet.

In both Kuhn poker and Adam poker the deck contains only three cards. A Jack, a Queen and a King. They have the property Jack $<$ Queen $<$ King. Observe that there is only one of each card. So two players can not have the same card.

For a complete background on Texas Hold 'em poker consult Collin Moshman's book on poker [2].

1.2 The game Kuhn poker

The game of Kuhn poker is a simplified version of poker. There are only two players p_1 and p_2. One is called the opener and the other is called the dealer. We assume that player p_1 is the first player to act (the opener), and player p_2 is the second to act (the dealer). The deck has three cards: a Jack, a Queen and a King. The players are dealt one card each, and the third is put aside. The players puts an ante of \$1 each into the pot. Then the first player can either bet \$1, or check. If p_1 bets player p_2 can either call the bet (paying another \$1, and the highest card wins the pot of \$4) or fold (player p_1 wins the pot of \$3). However if player p_1 checks, the roles are reversed and player p_2 can either check (showdown and the highest card wins the pot of \$2) or raise \$1. If he raises then player p_1 can call the bet (showdown for pot of \$4) or fold (player p_2 wins the pot of \$2).

So it is similar to a single round pre-flop of the game Limit Texas hold 'em with only three cards. Figure (1.2) shows the diagram of the full game.

The result of the analysis from [7] shows that the opener (first to act) has several optimal strategies and the dealer (second to act) has one optimal strategy which guarantees a positive expected value. To clarify, this means that the player who acts last has an advantage.

The opener has three decisions to make. He must decide how often to raise with the Jack (r_1), how often to call with the Queen (r_2) and how often to raise with the King (r_3). The dealer then has to decide how often to raise with the Jack when the opener has checked (q_1) and how often to call a raise holding the Queen (q_2). Those are the only decisions to be made. All other options are called "stupid mistakes" because they are never beneficial. For example if

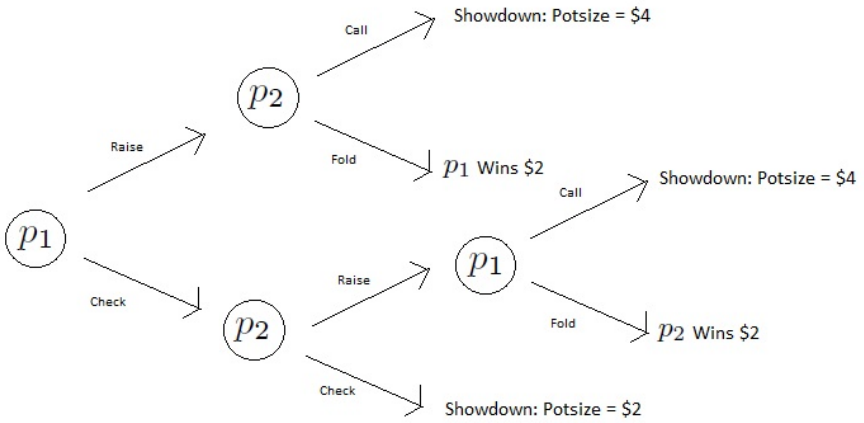


Figure 1: Diagram of the game Kuhn Poker.

the dealer holds the Jack, and the opener raises, then calling is guaranteed to result in a loss and is therefore excluded from the possible actions. All "Stupid Mistakes" are described in the article [7].

The expected value of the opener then is:

$$\frac{1}{6}[r_1(1-3q_2)+r_2(3q_1-1)+r_3(q_2-q_1)-q_1] \tag{1}$$

or equivalently

$$\frac{1}{6}[q_1(3r_2-r_3-1)+q_2(r_3-3r_1)+(r_1-r_2)]. \tag{2}$$

By looking at equation (1) we see that when $q_1 = q_2 = \frac{1}{3}$ the expected value becomes $\frac{-1}{18}$ and it does not depend on the opener's choices of r_1, r_2, or r_3. If the dealer deviates from this particular strategy, he allows for the opener to increase his expected value above $\frac{-1}{18}$, by choosing the appropriate counter-strategy. From equation (2) we can find a strategy for the opener which also has the property of being "indifferent" to the values of q_1 and q_2. The opener can choose r_3 arbitrarily and then pick

$$r_1 = \frac{1}{3}r_3$$

and

$$r_2 = \frac{1}{3}r_3 + \frac{1}{3}.$$

In this case the expected value becomes $\frac{-1}{18}$ no matter what strategy the dealer plays.

1.3 The game Adam poker

I have defined another version of poker which I have named Adam poker, where no optimal strategy exist.

Adam poker is also played with three cards. A Jack, a Queen and a King, with the property Jack < Queen < King. The game has only two players, p_1 and p_2. The game starts with one card being dealt to each player. Then each player pays the ante of \$1, making the total pot \$2. Then each player has two decisions, and they are carried out simultaneously. They can either check or raise. If both player check there will be a showdown, where the player with the highest card wins the pot of \$2. If both player raises, they put in \$1 each and the pot is then \$4, and the player with the highest card wins the pot of \$4. If one player checks, and one player raises, each take back what they put in.

If one player wins a pot of \$2 I will say that the player won \$1. The reason for this is that he won \$1 that was not his to start with, not \$2. And of course if a player wins a pot of \$4 I will say that the player won \$2 because that is how much the other player put in. The same is true for losing \$1 or \$2.

The game is somewhat simpler than Kuhn poker but it captures the essence of real Texas Hold'em which is important. Both Adam poker and Texas Hold'em are games of incomplete information, no optimal strategy exist and some type of bluffing is possible.

1.4 Introduction to game theory

Game theory is a branch of mathematics that deals with the analysis of conflict and cooperation between intelligent and rational players. Game theory is considered to have been introduced by John von Neumann (1903-1957) and Oskar Morgenstern (1902-1977) in the publication *The Theory of Games and Economic Behavior* in 1944 [3]. It described how mathematics could be used to analyze conflicts and decision making. By rational players one means that every player is assumed to act in their own self-interest [4].

The prisoner's dilemma is a classical example of a game analyzed in game theory [5]. The game can be described as follows. Two persons, A and B, have been imprisoned because the police suspect that they have committed a crime. They are held in isolated cells and the two prisoners have no way of communicating with each other. Now each prisoner is given the opportunity to betray the other prisoner by blaming the other prisoner, or they can stay silent. The possible outcomes are:

1. If both prisoners betray each other they both serve 6 months in prison.
2. If one betrays and one stays silent, the one betraying is set free and the one staying silent is sentenced to 9 months in prison.
3. If both players stay silent both of them will serve only 1 month in prison.

The payoff matrix for the game is

$$\begin{array}{c} \\ \text{Betray} \\ \text{Stay silent} \end{array} \begin{array}{c} \begin{array}{cc} \text{Betray} & \text{Stay silent} \end{array} \\ \begin{pmatrix} (-6,-6) & (0,-9) \\ (-9,0) & (-1,-1) \end{pmatrix} \end{array}$$

Both players are of course aware of the rules but unaware of the other player's move.

In this example it is assumed that both players are rational in the sense that they want to maximize their own payoff. We can see that betraying is the strategy every player should pick, since regardless of what the opponent does, picking the strategy "betray" is always beneficial. This is called Nash Equilibrium. A set of strategies is called a Nash Equilibrium if no player benefits from changing strategy given that the other players keep their current strategy constant.

Evolutionary game theory is the application of game theory to populations exposed to evolution, and is mainly used in biology. It focuses more on how the frequency of a strategy changes in a population of competing individuals, and has proven very useful in predicting when cooperation between individuals will take place. In the context of evolutionary game theory it is no longer needed to assume that the individuals or players are rational. Instead individuals who adopt a strategy which gives them a higher than average fitness will increase and individuals adopting a strategy which give them a fitness lower than average will decrease in the population. Here fitness can be interpreted as the probability of reproducing or the rate of reproduction. Having a high fitness implies that the individual will have a high probability of reproducing and hence to pass its strategy forward to the next generation. The success of a strategy is not only dependent on how good it is by itself, but rather how it works in a competitive population of other strategies.

The lifecycle for individuals in an evolutionary game is described as follows [6]:

1. Individuals form groups of size N. Individuals are distributed at random between groups.

2. Each individual has a strategy s_i. The payoff of each individual i is determined by its own strategy and the strategy of the other individuals. No reproduction occurs while the individuals are within the group.

3. The individuals leave the group and a law of selection is applied. Each strategy's contribution to the next generation is proportional to its fecundity relative to the average fecundity of the whole population. This contribution is known as the individual's fitness.

If we pick the group size $N = 2$, and define two strategies we get a payoff matrix like the one from the prisoner's dilemma. We now use the prisoner's dilemma to illustrate the basic principles of evolutionary game theory.

We assume that we have an infinite population of players, and each player either has the strategy betray or stay silent. We denote the proportion of the

population playing betray as x, and the proportion of the population playing stay silent with $(1-x)$. When a player with the strategy betray meets another one with betray we see that the outcome is -6 by looking at the payoff matrix, and when a betrayer meets a silent player the payoff is 0. Now since x is the probability of meeting a player with strategy betray, and $(1-x)$ is to meet a player with strategy stay silent, the fitness for a betrayer is

$$f_{\text{betray}} = -6x + 0(1-x) = -6x.$$

We apply the same argument to a silent player and find that its fitness is

$$f_{\text{stay silent}} = -9x - 1(1-x) = -8x - 1$$

The average payoff of the whole population is

$$\begin{aligned} \bar{F} &= x f_{\text{betray}} + (1-x) f_{\text{stay silent}} \\ &= (1-x)(-8x-1). \end{aligned}$$

We can then describe the rate of change of the proportion of betrayers in the population with the following expression known as the replicator equation [6]:

$$\begin{aligned} \frac{dx}{dt} &= x(\text{payoff of a betrayer - average payoff of population}) \qquad (3) \\ &= -x(2x+1)(x-1) = g(x) \end{aligned}$$

This equation has the roots $x = \{-1/2, 0, 1\}$. Since the proportion cannot be negative, only 0 and 1 are relevant to investigate. This corresponds to a population consisting of only silent players ($x = 0$) and only betrayers ($x = 1$). Those are the steady states, and we can determine their stability by differentiating $g(x)$ and evaluating it at the points 0 and 1. This gives

$$g'(x) = -6x^2 + 2x + 1,$$

and thus

$$g'(0) = 1, \;\; g'(1) = -3.$$

Since $g'(1) < 0$ the state at x = 1 is said to be evolutionary stable. It means that it cannot be invaded by a small amount of individuals with some other strategy. At the same time, $g'(0) > 0$, so a state where all players are silent players is an unstable state. If the replicator equations depends on several variables, differentiating g corresponds to finding the eigenvalues of the Jacobian matrix of the function g.

2 Analysis

2.1 Combinatorial analysis

In this section I will do an analysis which is similar to the one carried out in [7]. It is a straightforward combinatorial analysis to determine which play or strategy will be most beneficial. I will do the analysis for player p_1 but since there is no difference between p_1 and p_2 in Adam poker, it is identical to the analysis for player p_2. I will go through the three cases of cards for p_1 namely Jack, Queen and King.

The game of Adam poker has a payoff matrix

$$\begin{array}{c} \\ Ch \\ R \end{array} \begin{array}{c} \begin{array}{cc} Ch & R \end{array} \\ \begin{pmatrix} (1,-1) & (0,0) \\ (0,0) & (2,-2) \end{pmatrix} \end{array} \tag{4}$$

in case the row player has a higher card than the column player, and

$$\begin{array}{c} \\ Ch \\ R \end{array} \begin{array}{c} \begin{array}{cc} Ch & R \end{array} \\ \begin{pmatrix} (-1,1) & (0,0) \\ (0,0) & (-2,2) \end{pmatrix} \end{array} \tag{5}$$

if the row player has a lower card than the column player.

Let us start by investigating the case where the row player p_1 has the lowest card, Jack. Then the payoff matrix for the game is as in (5). This means that whenever both players do the same move there is a showdown and player p_1 having the lowest card loses. If both check he loses \$1 and if both raise he loses \$2. If one player checks and the other player raises both players win \$0. In this case we can easily see that the best strategy for p_1 would be to always do the opposite of what p_2 does, resulting in no loss. The best strategy for opponent p_2 on the other hand is to always do the same as p_1 which results in a showdown.

Now let us define C_1 as the probability that player p_1 checks and $(1 - C_1)$ that he raises, C_2 that player p_2 checks, and $(1 - C_2)$ that he raises. The expected value for player p_1 becomes

$$E_1 = -1(C_1C_2) - 2(1 - C_1)(1 - C_2).$$

If both players check (C_1C_2) player p_1 loses \$1, and if both raise $((1 - C_1)(1 - C_2))$ he loses \$2. If they do different moves the win is 0 and is therefore excluded. After some algebra we arrive at

$$E_1 = C_1(2 - 3C_2) + 2C_2 - 2. \tag{6}$$

To maximize the expected value player p_1 should pick $C_1 = 1$ when the parenthesis is positive, that is when $C_2 < \frac{2}{3}$, and $C_1 = 0$ when $C_2 > \frac{2}{3}$. In the first case the expected value becomes $-C_2$ and in the latter case it becomes $2C_2 - 2$. In case player p_2 picks $C_2 = \frac{2}{3}$ the expected value becomes $\frac{-2}{3}$ independent of C_1.

Note that this result holds when p_1 has Jack, and p_2 has either Queen or King.

Next we look at the case when p_1 has the highest card, King, and player p_2 has either Jack or Queen. It is basically the opposite of the previous result. Using the payoff matrix in (4) we get the expected value

$$E_2 = 1(C_1C_2) + 2(1 - C_1)(1 - C_2)$$

which simplifies to

$$E_2 = C_1(3C_2 - 2) - 2C_2 + 2. \tag{7}$$

To maximize the expected value p_1 should pick $C_1 = 1$ when the parenthesis is positive, that is when $C_2 > \frac{2}{3}$, and $C_1 = 0$ when the parenthesis is negative, which happens when $C_2 < \frac{2}{3}$. In the first case the expected value becomes C_2 and in the latter case $2 - 2C_2$. If $C_2 = \frac{2}{3}$ the expected value becomes $\frac{2}{3}$ independent of C_1.

Now we look at the case where p_1 has Queen, and p_2 either Jack or King. We now define $C_{2,J}$ and $C_{2,K}$ as the probability that p_2 checks having Jack and King respectively. We then apply the first result to the case where p_2 has King, and the second result to the case where p_2 has Jack. We multiply both by $\frac{1}{2}$ since both cases have probability $\frac{1}{2}$ to occur. The expected value is

$$E_3 = \frac{1}{2}(C_1(C_{2,J}) + 2(1 - C_1)(1 - C_{2,J})$$

$$+\frac{1}{2}(-C_1(C_{2,K}) - 2(1 - C_1)(1 - C_{2,K})),$$

which simplifies to

$$E_3 = \frac{(C_{2,J} - C_{2,K})(3C_1 - 2)}{2}. \tag{8}$$

Just as before we want to maximize the expected value, by choosing $C_1 = 1$ when the left parenthesis is positive, which happens when $C_{2,J} > C_{2,K}$ and $C_1 = 0$ when the left parenthesis is negative which happens when $C_{2,J} < C_{2,K}$. In the first case the expected value becomes $C_{2,K} - C_{2,J}$ and in the second case $\frac{C_{2,J} - C_{2,K}}{2}$. If $C_{2,J} = C_{2,K}$ the expected value becomes 0, independent of C_1.

This is intuitive because of the fact that when p_1 is holding the Queen, the best play is to imitate the play done by the opponent when he is holding the lower card, Jack. When the opponent is checking with the King more often than with the Jack, player p_1 instead wants to raise to avoid meeting the King in a showdown. To make an actual strategy of the three results from equations (6), (7) and (8), we should switch C_2 in equation (6) and (7) for $\frac{C_{2,Q} + C_{2,K}}{2}$ when p_1 has the Jack, and $\frac{C_{2,J} + C_{2,Q}}{2}$ when p_1 has the King.

2.2 Analysis using evolutionary game theory

2.2.1 A special case

We now use a different approach to our problem. We start by studying the simple case where we imagine that there is a population of players who always have a Jack. The other part of the population always has a Queen or a King. So every player has either a Jack or any higher card (Queen or King), and also a strategy (check or raise). So we define x to be the proportion of the population holding the Jack with the strategy check, and $(1-x)$ is the proportion of the population holding the Jack and playing the strategy raise. Here $x \in [0,1]$. In a similar fashion we define y to be the proportion of the population having a higher card (Queen or King) and playing the strategy check, and $(1-y)$ playing raise, with $y \in [0,1]$. The payoff matrix is the same as before:

$$\begin{array}{cc} & \begin{array}{cc} Ch & R \end{array} \\ \begin{array}{c} Ch \\ R \end{array} & \begin{pmatrix} (-1,1) & (0,0) \\ (0,0) & (-2,2) \end{pmatrix}. \end{array}$$

We now define U to be the proportion of the population having an undercard (Jack) and O to be the proportion of the population having an overcard (Queen or King). The functions for the expected payoff (fitness functions) for each proportion U and O with the action either check or raise are:

$$\begin{aligned} f_U^C(x,y) &= -y \\ f_U^R(x,y) &= -2(1-y) \\ f_O^C(x,y) &= x \\ f_O^R(x,y) &= 2(1-x) \end{aligned}$$

where for example f_U^C is the fitness function for the proportion U checking, f_O^R is the fitness function for the propotion O raising and so on.

The fitness function can be derived from the payoff matrix and the probability (proportion of population) of meeting a player with the corresponding strategy. Now we want to calcualte the average payoff for all players with the lowest card, and the average payoff for all players having an overcard. The average payoff for each part is

$$\begin{aligned} f_U^A(y) &= x f_x^C + (1-x) f_x^R \\ f_O^A(x) &= y f_y^C + (1-y) f_y^R \end{aligned}$$

and in our case the equations become

$$f_U^A(y) = -xy - 2(1-x)(1-y)$$
$$f_O^A(x) = xy + 2(1-y)(1-x).$$

To clarify this, f_U^A is the average fitness for all players with an undercard (Jack), and f_O^A is the average fitness for all players with an overcard (Queen or King).

Now we can write the expression for the rate of change of x, and similarly the rate of change of y, as

$$\begin{aligned}\frac{dx}{dt} &= x(f_x^C - f_U^A)\\ &= x(-y + xy + 2(1-x)(1-y))\\ &= x(1-x)(2-3y)\\ \frac{dy}{dt} &= y(f_y^C - f_O^A)\\ &= y(x - xy - 2(1-y)(1-x))\\ &= y(1-y)(3x-2)\end{aligned}$$

To motivate the equations we note that the rate of change of x is positive if the fitness of strategy check (f_x^C) is better than the average fitness (f_U^A) of all players with an undercard. If it is, the expression is positive. If f_x^C is less than f_U^A, it means that checking has lower than average fitness (the rate of change is negative) and hence the amount of x's in the population decreases.

Let us denote $f(x) := \frac{dx}{dt}$ and $g(x) := \frac{dy}{dt}$.

We now have a system of two ordinary differential equations, and we want to find the critical points. We do so by setting $\frac{dx}{dt} = \frac{dy}{dt} = 0$. This gives us the five points $(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$ and $(\frac{2}{3}, \frac{2}{3})$. We can see some resemblance with results (6) and (7) from the previous section.

To determine the stability we study the Jacobian matrix

$$J = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix},$$

where f and g are as above.

Inserting our functions we get

$$J = \begin{pmatrix} (1-2x)(2-3y) & -3(x-x^2) \\ 3(y-y^2) & (1-2y)(3x-2) \end{pmatrix}.$$

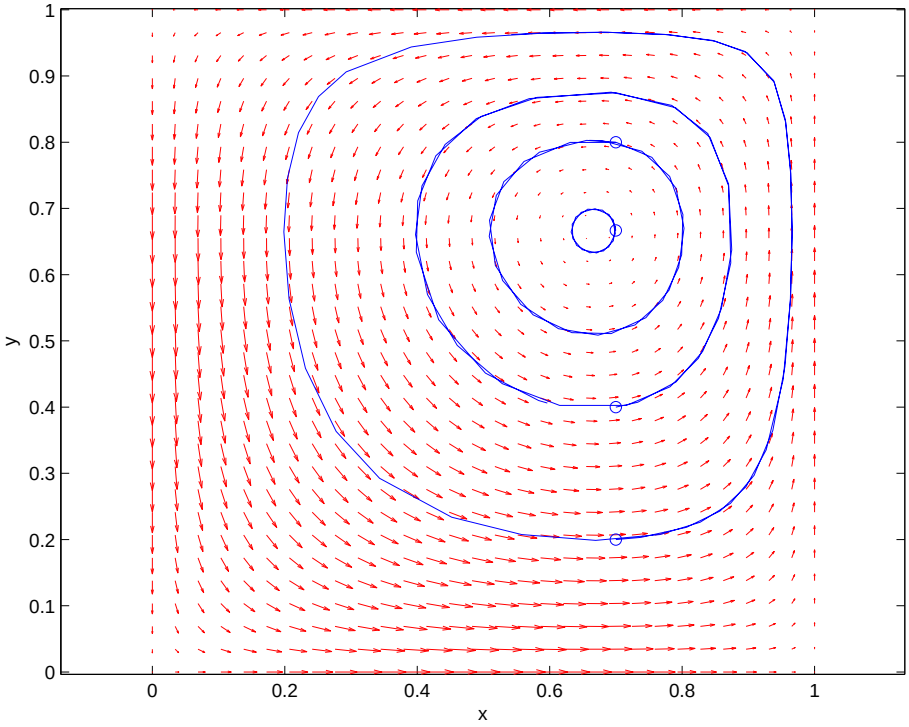


Figure 2: Phase portrait of the system of ODEs.

At the point $(\frac{2}{3}, \frac{2}{3})$ the eigenvalues are $\pm i\frac{2}{3}$, and at all other points the Jacobian becomes a diagonal matrix with one positive and one negative real eigenvalue which correnspponds to a saddle point. Having two purely complex eigenvalues corresponds to a center [1], which is confirmed by the phase portrait.

So the proportions oscillate, and the period seems to depend on the initial values.

This was just the special case where every individual in the population either had the lowest card and the other player had an overcard. Now we want to take the next step and include the case where every player can have either Jack, Queen or King.

2.2.2 The full analysis

In this section we do the full analysis. We start by defining the strategies s_i, then we define x_i as the proportion of the population having the strategy s_i. It turns out that there are 8 different strategies in total. We proceed as before by defining the fitness function for each strategy and denote it by $F_i(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \dots x_8)$. When we have all the functions we calculate the average fitness function $\bar{F} = \sum_{i=1}^{8} x_i(F_i)$. Then we only have one thing left, namely the

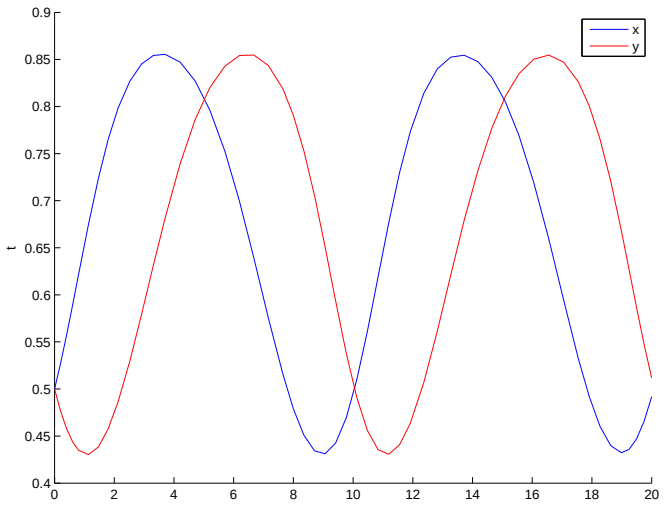


Figure 3: Plot of x and y with respect to t with initial values $x = 0.5$, $y = 0.5$.

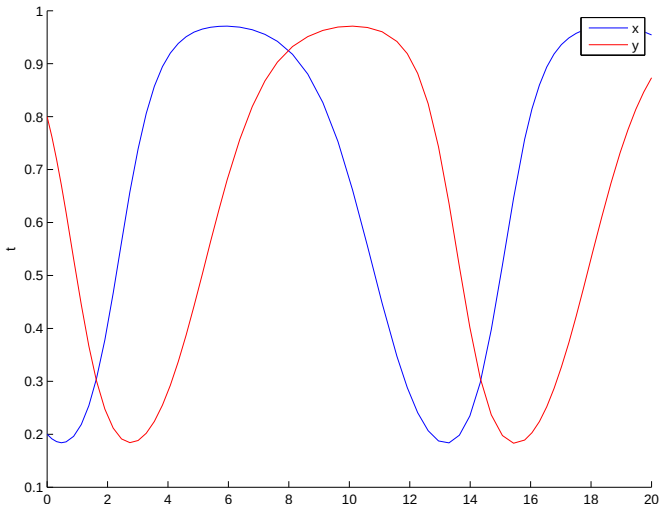


Figure 4: Plot of x and y with respect to t with initial values $x = 0.2$, $y = 0.8$.

replicator equation for each proportion x_i. It is defined as $\frac{dx_i}{dt} = x_i(F_i - \bar{F})$. The calculations are very long and I have used MATLAB to define all the functions. After that I solved the replicator equations and plotted the result for a few different initial values.

A strategy for a player is a set of actions the player will take with each card. The actions can be either check or raise, and we denote check with a 0 and raise with a 1. And since there are three cards a strategy will be of the form (a, b, c) where a, b and c is either 0 or 1. All eight strategies are

- $s_1 = (0, 0, 0)$
- $s_2 = (1, 0, 0)$
- $s_3 = (0, 1, 0)$
- $s_4 = (0, 0, 1)$
- $s_5 = (1, 1, 0)$
- $s_6 = (1, 0, 1)$
- $s_7 = (0, 1, 1)$
- $s_8 = (1, 1, 1)$

Example 1. *The strategy* $(1, 1, 0)$ *means that the player will raise with Jack and Queen, and check with King.*

The fitness function F_i for the strategy s_i is

$$F_i(\mathbf{x}) = \sum_{j=1}^{8} x_i(\text{Payoff for } s_i \text{ meeting } s_j),$$

where $\mathbf{x} = (x_1, x_2, \ldots x_8)$.

To calculate the fitness we need to take all the cards into consideration. Here is an example how it is done.

Example 2. *What is the payoff when* $s_4 = (0, 0, 1)$ *meets* $s_6 = (1, 0, 1)$*? We must calculate the payoff for all six possible card combinations:* (J, Q)*,* (J, K)*,* (Q, J)*,* (Q, K)*,* (K, J)*,* (K, Q)*. Each has the probability* $\frac{1}{6}$ *of occurring, and using the order just mentioned we get the expected win for* s_1*:* $\frac{1}{6}(-1 + 0 + 0 + 0 + 2 + 0) = \frac{1}{6}$ *by using the payoff matrices in (4) and (5). Which matrix is used depends on the player's card. The same is applied for each and every opponent, and that gives us the function* F_4 *for a player with strategy* s_4*.*

Repeating the calculations for every strategy we get all eight fitness functions:

- $F_1(\mathbf{x}) = (\frac{-1}{3})x_2 + (\frac{1}{3})x_4 + (\frac{-1}{3})x_5 + (\frac{1}{3})x_7$

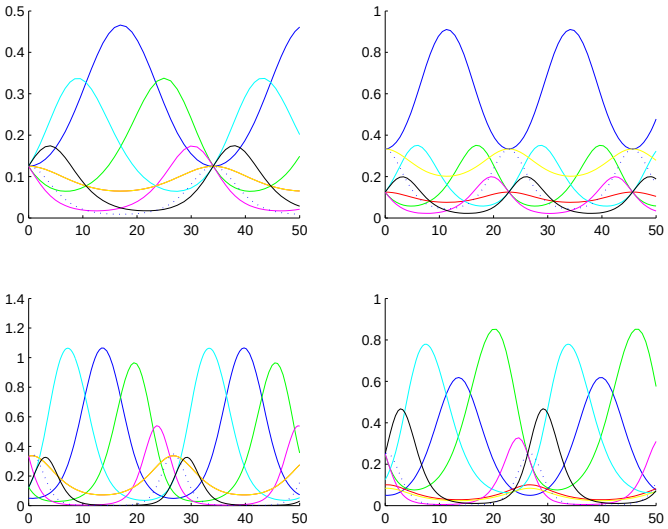


Figure 5: Solution to the replicator equations with different initial values, with number of generations on the x-axis and proportion of the population on the y-axis.

- $F_2(\mathbf{x}) = (\frac{1}{3})x_1 + (\frac{-1}{6})x_3 + (\frac{1}{6})x_4 + (\frac{-1}{2})x_5 + (\frac{-1}{6})x_6 + (\frac{-1}{3})x_7 + (\frac{-2}{3})x_8$
- $F_3(\mathbf{x}) = (\frac{1}{6})x_2 + (\frac{-1}{6})x_4 + (\frac{1}{6})x_5 + (\frac{-1}{6})x_7$
- $F_4(\mathbf{x}) = (\frac{-1}{3})x_1 + (\frac{-1}{6})x_2 + (\frac{1}{6})x_3 + (\frac{1}{3})x_5 + (\frac{1}{6})x_6 + (\frac{1}{2})x_7 + (\frac{2}{3})x_8$
- $F_5(\mathbf{x}) = (\frac{1}{3})x_1 + (\frac{1}{2})x_2 + (\frac{-1}{6})x_3 + (\frac{-1}{3})x_4 + (\frac{-1}{6})x_6 + (\frac{-5}{6})x_7 + (\frac{-2}{3})x_8$
- $F_6(\mathbf{x}) = (\frac{1}{6})x_2 + (\frac{-1}{6})x_4 + (\frac{1}{6})x_5 + (\frac{-1}{6})x_7$
- $F_7(\mathbf{x}) = (\frac{-1}{3})x_1 + (\frac{1}{3})x_2 + (\frac{1}{6})x_3 + (\frac{-1}{2})x_4 + (\frac{5}{6})x_5 + (\frac{1}{6})x_6 + (\frac{2}{3})x_8$
- $F_8(\mathbf{x}) = (\frac{2}{3})x_2 + (\frac{-2}{3})x_4 + (\frac{2}{3})x_5 + (\frac{-2}{3})x_7$

The average fitness function $\bar{F}$ and the eight replicator equations $\frac{dx_i}{dt}$ are calculated, and solved using MATLAB. The result is seen in Figure 5.

The full MATLAB-code that I have used can be found in Appendix A.1, A.2 and A.3. Observe that the average fitness of the population is constantly 0. This is not a surprise since Adam poker is a zero-sum game i.e. one player's win is another player's loss. Therefore the average fitness cannot change.

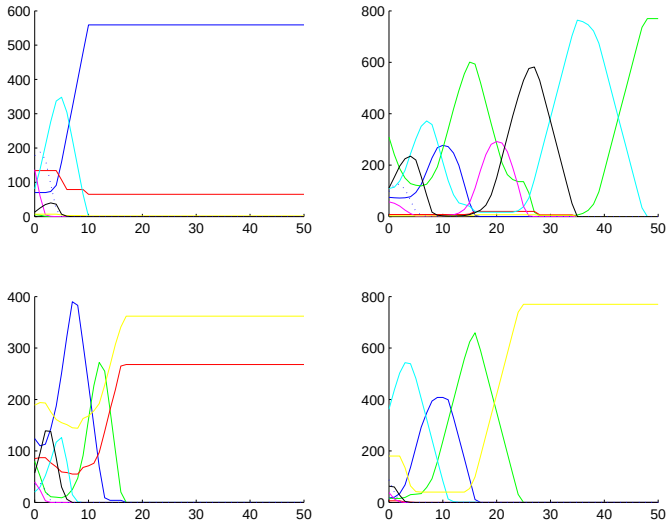


Figure 6: Four plots of the simulation with randomized initial values, number generations on the x-axis and number of individuals on the y-axis.

2.3 Simulation

Now I have done a simulation of a population of poker playing individuals. Each individual has a fixed strategy s_i corresponding to the strategies described earlier. Then they are paired together 1-1 and play a game of Adam poker. Each player also has a score which is the accumulated win or loss after a number of matches (or games). After a number of matches the scores are evaluated. The 80 individuals with the lowest scores are erased, and the 80 individuals having the highest score are duplicated. That way the population size is held constant and a kind of selection occurs. We call one such cycle a generation. I have performed the simulations with both fixed initial number of inidividuals, and with randomized initial values. The full MATLAB code for the simulation can be found in Appendix A.4.

The result is similar to the deterministic model based on the replicator equations. The number of individuals oscillate like it did in the deterministic model, but now when we introduced random pairing it appears to disturb the "niceness" and results in the extinction of strategies. The plot of four simulations is seen in Figure 6.

In Figure 7 and 8 I have included a ninth strategy, $s_9 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, in the deterministic and stochastic model respectively. It is the strategy where the player is raising with probability $\frac{1}{3}$. It is motivated by the results from the combinatorial analysis in section 2.1. It is not a surprise to see that strategy s_9 is the constant strategy.

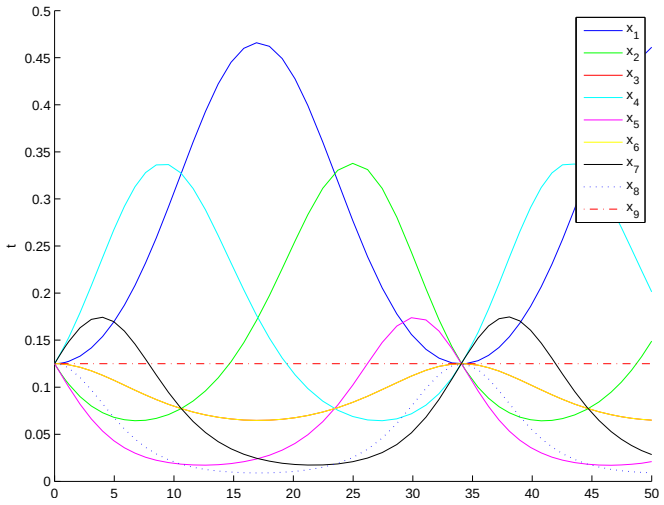


Figure 7: Solution to the replicator equations including the ninth strategy.

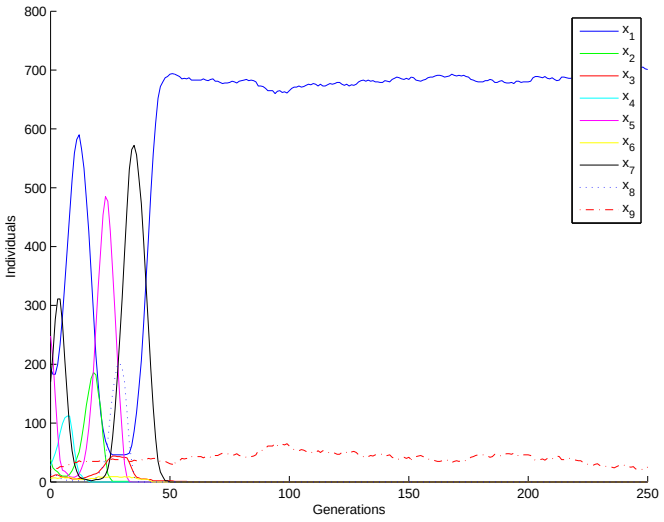


Figure 8: Simulation with randomized initial values, including the ninth strategy.

2.4 Finding the critical points

To find the steady states we need to find the critical points of the replicator equations and then we determine their stability.

To find the steady states we need to solve the system of equations

$$\begin{cases} \frac{dx_1}{dt} = 0 \\ \vdots \\ \frac{dx_8}{dt} = 0 \end{cases},$$

with $x_1 + \ldots x_8 = 1$, and all the x_i's positive. The replicator equations are $\frac{dx_i}{dt} = x_i(F_i - \bar{F})$, and in this game the average fitness of the population is 0. This can be shown by basic algebra after simplification of $\bar{F} = \sum_{i=1}^{8} x_i F_i$. So we are left with $\frac{dx_i}{dt} = x_i F_i$. This is solved using MATLAB and it results in 19 critical points. The first eight are the points corresponding to all proportions x_i are 0 except for one. The other points are:

$$c_1 = \left(\frac{1}{3}, 0, 0, 0, 0, \frac{2}{3}, 0, 0\right)$$

$$c_2 = \left(\frac{2}{3}, 0, 0, 0, 0, 0, 0, \frac{1}{3}\right)$$

$$c_3 = \left(\frac{1}{3}, 0, \frac{2}{3}, 0, 0, 0, 0, 0\right)$$

$$c_4 = \left(\frac{1}{3}, 0, 0\frac{1}{3}, \frac{1}{3}, 0, 0, 0\right)$$

$$c_5 = \left(0, \frac{1}{2}, 0, \frac{1}{3}, 0, 0, \frac{1}{6}, 0\right)$$

$$c_6 = \left(0, \frac{1}{3}, 0, \frac{1}{2}, \frac{1}{6}, 0, 0, 0\right)$$

$$c_7 = \left(\frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0, \frac{1}{3}, 0\right)$$

$$c_8 = \left(0, \frac{1}{3}, 0, \frac{1}{3}, 0, \frac{1}{3}, 0, 0\right)$$

$$c_9 = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0\right)$$

$$c_{10} = \left(0, \frac{4}{9}, 0, \frac{4}{9}, 0, 0, 0, \frac{1}{9}\right)$$

$$c_{11} = \left(\frac{5}{9}, 0, 0, 0, \frac{2}{9}, \frac{2}{9}, 0, 0\right)$$

Using the Jacobian matrix and substituting each critical point we get the following result:

- For the first eight critical points where only one strategy survives, the eigenvalues are all real, some positive and some negative, and the rest are zero.
- In the case where two strategies survive, all eigenvalues are zero
- When three strategies survive, two eigenvalues are purely complex and the rest are zero.

2.5 Why some strategies can coexist

We once again have a look at the fitness matrix containing the expected value of each strategy:

- $F_1(\mathbf{x}) = (\frac{-1}{3})x_2 + (\frac{1}{3})x_4 + (\frac{-1}{3})x_5 + (\frac{1}{3})x_7$
- $F_2(\mathbf{x}) = (\frac{1}{3})x_1 + (\frac{-1}{6})x_3 + (\frac{1}{6})x_4 + (\frac{-1}{2})x_5 + (\frac{-1}{6})x_6 + (\frac{-1}{3})x_7 + (\frac{-2}{3})x_8$
- $F_3(\mathbf{x}) = (\frac{1}{6})x_2 + (\frac{-1}{6})x_4 + (\frac{1}{6})x_5 + (\frac{-1}{6})x_7$
- $F_4(\mathbf{x}) = (\frac{-1}{3})x_1 + (\frac{-1}{6})x_2 + (\frac{1}{6})x_3 + (\frac{1}{3})x_5 + (\frac{1}{6})x_6 + (\frac{1}{2})x_7 + (\frac{2}{3})x_8$
- $F_5(\mathbf{x}) = (\frac{1}{3})x_1 + (\frac{1}{2})x_2 + (\frac{-1}{6})x_3 + (\frac{-1}{3})x_4 + (\frac{-1}{6})x_6 + (\frac{-5}{6})x_7 + (\frac{-2}{3})x_8$
- $F_6(\mathbf{x}) = (\frac{1}{6})x_2 + (\frac{-1}{6})x_4 + (\frac{1}{6})x_5 + (\frac{-1}{6})x_7$
- $F_7(\mathbf{x}) = (\frac{-1}{3})x_1 + (\frac{1}{3})x_2 + (\frac{1}{6})x_3 + (\frac{-1}{2})x_4 + (\frac{5}{6})x_5 + (\frac{1}{6})x_6 + (\frac{2}{3})x_8$
- $F_8(\mathbf{x}) = (\frac{2}{3})x_2 + (\frac{-2}{3})x_4 + (\frac{2}{3})x_5 + (\frac{-2}{3})x_7$

If we look at the fitness function for strategy s_1 we have

$$F_1(\mathbf{x}) = (\frac{-1}{3})x_2 + (\frac{1}{3})x_4 + (\frac{-1}{3})x_5 + (\frac{1}{3})x_7$$

and we see that its expected value against other players with strategy s_1 is 0. That is also the case against s_3, s_6 and s_8. This means that they can coexist.

It is easily seen that the four strategies which can coexist are s_1, s_3, s_6 and s_8. They all have expected value 0 against each other. In comparison, the other four strategies s_2, s_4, s_5 and s_7 have a non-zero expected value against all other strategies except against themselves. This does not imply that they cannot coexist, but because every critical point has at least one eigenvalue which is non-negative none of them are asymptotically stable, and therefore they cannot coexist. They must either win or lose if they play a large number of games.

Another very important observation is that strategy s_3 and s_6 behave identically. The strategies are not identical, which can be seen by looking at how they are defined, but their fitness functions are identical. That means that they behave in the same way in a population of poker playing individuals. Below is a table containing the fitness functions, but instead of numerical values it contains only a plus or minus depending on the sign of the entry in the fitness function.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	-	0	+	-	0	+	0
x_2	+	0	-	+	-	-	-	-
x_3	0	+	0	-	+	0	-	0
x_4	-	-	+	0	+	+	+	+
x_5	+	+	-	-	0	-	-	-
x_6	0	+	0	-	+	0	-	0
x_7	-	+	+	-	+	+	0	+
x_8	0	+	0	-	+	0	-	0

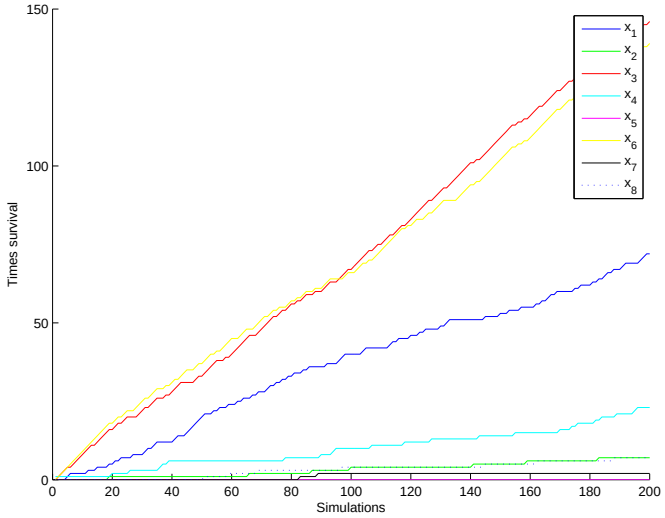


Figure 9: Total number of times each strategy survived.

2.6 How often each strategy survives

In this section I present the results of doing repeated simulations and counting how many times each strategy survived. I repeated 200 simulations and counted how many times each strategy survived in total in Figure 9, and in what combinations they survived in Figure 10.

3 Conclusion

The goal of my thesis was to define a version of poker which did not have an optimal strategy. Adam poker has that property. If I know my opponenet's strategy, there is at least one counterstrategy, and the same is true for my opponent. So in a real game of Adam poker the goal would be to determine the opponent's strategy, and then to pick a counterstrategy.

I used both a deterministic model consisting of the replicator equations, and a stochastic model which simulated a real population of poker playing individuals. They yielded similar results, with the exception that in the deterministic model no strategy became extinct. In the deterministic model we see that all eight strategies can in fact survive and coexist.

During the analysis I also realized several interesting properties of the game. Strategy s_3 and s_6 are not identical, but they behave identically in a population of poker playing individuals, because they have the exact same fitness function. Those are also the overall best strategies in a mixed population.

There are four strategies, s_1, s_3, s_6 and s_8 which can coexist. Their fitness functions have value zero against each other, which means that as they play

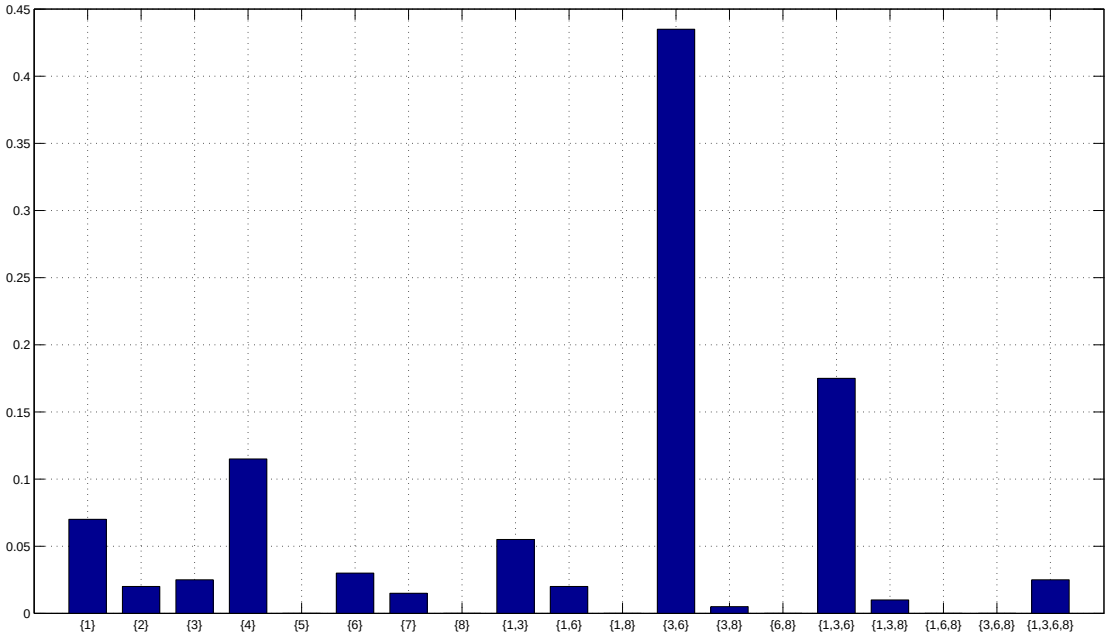


Figure 10: How often the surviving strategies survived in different combinations. The x-axis shows strategy combinations and the y-axis shows the percentage of all simulations

many games of Adam poker, their expected win or loss is 0. All other strategies have the property that they cannot survive together with any other strategy. In their case the fitness functions are non-zero against all other strategies, and no critical point is asymptotically stable. But of course they can survive by themselves.

References

[1] Morris W. Hirsch, Stephen Smale, and Robert L. Devaney. *Differential Equations, Dynamical Systems, and An Introduction to Chaos*. Elsevier Science, second edition, 2004.

[2] Collin Moshman. *Heads-Up No-Limit Hold 'em: Expert Advice for Winning Heads-Up Poker Matches*. Two Plus Two Publishing, 2008.

[3] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 60th anniversary commemorative edition, 2007.

[4] Martin A. Nowak. *Evolutionary Dynamics*. The Belknap Press, 2006.

[5] Graham Romp. *Game Theory Introduction and Applications*. Oxford University Press, 1997.

[6] David J. T. Sumpter. *Collective Animal Behavior*. Princeton University Press, 2010.

[7] Jason Swanson. Game theory and poker. `http://math.swansonsite.com/instructional/game_theory.pdf`, April 2005.

A MATLAB code

This sections contains the basic code I have used. Some parts of the simulations have been based on this code with minor adjustments.

A.1 Fitness

```
function [Yout] = Fitness(Y)
% Contains the fitness matrix. Input: row vector of length 9.
% Returns a column vector of length 9.
% The function has support for the ninth strategy.
Yout = [(-1/3)*Y(2) + (1/3)*Y(4) - (1/3)*Y(5) + (1/3)*Y(7);
    (1/3)*Y(1) - (1/6)*Y(3) + (1/6)*Y(4) - (1/2)*Y(5) - (1/6)*Y(6) -
        (1/3)*Y(7) - (2/3)*Y(8);
    (1/6)*Y(2) - (1/6)*Y(4) + (1/6)*Y(5) - (1/6)*Y(7);
    (-1/3)*Y(1) - (1/6)*Y(2) + (1/6)*Y(3) + (1/3)*Y(5) + (1/6)*Y(6)
        + (1/2)*Y(7) + (2/3)*Y(8);
    (1/3)*Y(1) + (1/2)*Y(2) - (1/6)*Y(3) - (1/3)*Y(4) - (1/6)*Y(6) -
        (5/6)*Y(7) - (2/3)*Y(8);
    (1/6)*Y(2) - (1/6)*Y(4) + (1/6)*Y(5) - (1/6)*Y(7);
    (-1/3)*Y(1) + (1/3)*Y(2) + (1/6)*Y(3) - (1/2)*Y(4) + (5/6)*Y(5)
        + (1/6)*Y(6) + (2/3)*Y(8);
    (2/3)*Y(2) - (2/3)*Y(4) + (2/3)*Y(5) - (2/3)*Y(7)
    0*Y(9)];
end
end
```

A.2 Replicator

```
function [dxdt] = Replicator(t, Y)
%Contains the replicator equations. It has support for the ninth
    strategy.
%Observe that the average fitness (AvFi) is 0.
format long
Fit = Fitness(Y);
dxdt = [Y(1)*(Fit(1)-AvFi);
    Y(2)*(Fit(2)-AvFi);
    Y(3)*(Fit(3)-AvFi);
    Y(4)*(Fit(4)-AvFi);
    Y(5)*(Fit(5)-AvFi);
    Y(6)*(Fit(6)-AvFi);
    Y(7)*(Fit(7)-AvFi);
    Y(8)*(Fit(8)-AvFi)
    Y(9)*(Fit(9)-AvFi)];
end
```

A.3 Calculation

```
%Script that solves and plots the replicator equations using ODE45.
tlength = 10;
t = [0:0.1:tlength];
initial = [1 1 1 1 1 1 1 1 1]

figure
hold on
[ts, ys] = ode45(@Replicator, [0, tlength], initial);
plot(ts, ys(:,1));
plot(ts, ys(:,2),'g');
plot(ts, ys(:,3),'r');
plot(ts, ys(:,4),'c');
plot(ts, ys(:,5),'m');
plot(ts, ys(:,6),'y');
plot(ts, ys(:,7),'k');
plot(ts, ys(:,8),':');
%plot(ts, ys(:,9),'r-.'); % The ninth strategy is constant.
xlabel('time')
ylabel('individuals')
legend('x_1','x_2','x_3','x_4','x_5','x_6','x_7','x_8', 'x_9');
```

A.4 Playnonrand

```
function [P1WIN] = Playnonrand(P1Strat, P2Strat)
% Used to simulate a match of Adam Poker between the two strategies
% P1Strat and P2strat.
% Returns the win of the player with strategy P1Strat.
F = [0 -1/3 0 1/3 -1/3 0 1/3 0;
1/3 0 -1/6 1/6 -1/2 -1/6 -1/3 -2/3;
0 1/6 0 -1/6 1/6 0 -1/6 0;
-1/3 -1/6 1/6 0 1/3 1/6 1/2 2/3;
1/3 1/2 -1/6 -1/3 0 -1/6 -5/6 -2/3;
0 1/6 0 -1/6 1/6 0 -1/6 0;
-1/3 1/3 1/6 -1/2 5/6 1/6 0 2/3;
0 2/3 0 -2/3 2/3 0 -2/3 0];

P1WIN = F(P1Strat,P2Strat);
end
```

A.5 RandomSimulation

```
% Script used to simulate the population of poker playing
    individuals over a number of generations.
clear
%Random initial values are determined.
proportion = rand(1,8);
proportion = sort(proportion);

for i = 1:7
    j = 9-i;
```

```
    proportion(j) = proportion(j) - proportion(j-1);
end

for i = 1:8
    if(round(proportion(i)*800) == 0)
        numberOfIndividuals(i) = 1;
    else
numberOfIndividuals(i) = round(proportion(i)*800);
    end
end

%Create n*2 matrix M, where n is number of individuals.
% M = [strategy, score]. A row represents one individual.
numberOfIndividuals = cumsum(numberOfIndividuals);

for i = 1 : numberOfIndividuals(1)
    M(i,1) = 1;
end
for i = numberOfIndividuals(1):numberOfIndividuals(2);
    M(i,1) = 2;
end
for i = numberOfIndividuals(2):numberOfIndividuals(3);
    M(i,1) = 3;
end
for i = numberOfIndividuals(3):numberOfIndividuals(4);
    M(i,1) = 4;
end
for i = numberOfIndividuals(4):numberOfIndividuals(5);
    M(i,1) = 5;
end
for i = numberOfIndividuals(5):numberOfIndividuals(6);
    M(i,1) = 6;
end
for i = numberOfIndividuals(6):numberOfIndividuals(7);
    M(i,1) = 7;
end
for i = numberOfIndividuals(7):numberOfIndividuals(8);
    M(i,1) = 8;
end

%Make sure there are an even number of individuals.
%If it is not even, add another individual with strategy 8.
if(mod(size(M),2) == 1)
    M(size(M)+1, 1) = 8;
end

%Add the right column containing the scores (currently only zeros).
k = size(M,1);
right = zeros(k, 1);
M = [M right];

%Count how many individuals there are of each strategy.
for i = 1:8
count = find(M == i);
n = length(count);
X(1,i) = n;
end
```

```
66
67  %Number of generations and number of games played each generation.
68  gen = 50;
69  games = 10;
70
71  for j = 1:gen
72
73  for i = 1:games;
74
75  %Contains the indexes of all players.
76  NumArray = linspace(1, size(M,1), size(M,1));
77
78  %Now pair each element with another element 1-1 fashion:
79  for i = 1:size(M,1)/2
80
81  %The index (row) of P1.
82  firstindex = 1 + round(rand(1)*(length(NumArray) -1));
83  first = NumArray(firstindex);
84
85  %Remove the first from the array.
86  NumArray = NumArray(NumArray~=first);
87
88  %The index (row) of P2.
89  secondindex = 1 + round(rand(1)*(length(NumArray) -1));
90  second = NumArray(secondindex);
91
92  %Remove the second from the array.
93  NumArray = NumArray(NumArray~=second);
94
95  %Now let them play!
96  WIN = Playnonrand(M(first,1), M(second,1));
97
98  %Give them their scores after the game.
99  M(first,2) = M(first,2) + WIN;
100 M(second,2) = M(second,2) - WIN;
101 end
102
103 end
104
105 %Remove individuals.
106 for i = 1:80
107 Scores = M(:,2);
108 [C,I] = min(Scores);
109 Y = find(Scores <= C+0.00001);
110 Index = ceil(rand*length(Y));
111 I = Y(Index);
112
113 M(I,:) = [];
114 end
115
116 NewM = [];
117
118 %Add individuals.
119 for i = 1:80
120 Scores = M(:,2);
121 [C,I] = max(Scores);
122
```

```
123 Z = find(Scores >= C-0.00001);
124 Index = ceil(rand*length(Z));
125 I = Z(Index);
126
127 %To know what new individual to create.
128 Strat = M(I,1);
129 NewM = [NewM; Strat 0; Strat 0];
130
131 %Must delete the "winning one" so it does not get chosen again.
132 M(I,:) = [];
133 end
134
135 %Concatenate the old matrix M with the new rows.
136 M = [M; NewM];
137
138 %Reset the scores
139 M(:,2) = 0;
140 M = sort(M);
141
142 %Take the info for the plot in each stage.
143 for i = 1:8
144 count = find(M == i);
145 n = length(count);
146 X(j+1,i) = n;
147 end
148 end
```