SAINT-PETERSBURG STATE UNINERSITY

Department of Mathematical Games Theory and Statistical Decisions

«Admissible Forms of Cooperation in Multistage Games»

Master's thesis

Kostetskaya Aleksandra Gennadievna

Specialization 01.04.02

Applied Mathematics and Control Processes

Game Theory and Operations Research

Scientific adviser:

Doctor of physical-mat.sciences'

Prof. Petrosyan L.A.

Reviewer:

Candidate of physical-mat.sciences'

Kolabutin N.V.

Saint-Petersburg

2019

Content

Introduction

In this research, I will explore two games.

Both games consider the presence of the mental system of the players. The problem with the introduction of mental systems and extraneous signals - it is still a little studied problems. These problems are only gaining popularity.

The aim of this research is to study the influence of the mental system of the players and receive signals from the outside, on the possibility of cooperation.

The first game was taken from the [1] article, where two players give each other gifts. These gifts can have different values, and mood of players can be different at different stages of the game. It depends on what value the gift will be presented to the other player. Future interactions provide incentives for collaboration in a repeated game. I have a model in which players achieve cooperation in an intuitively plausible way. Players process information through the mental system-a set of psychological states and the transition function between states depending on observations. Observations are signals that each player receives at each stage of the game. Players limit their attention to a relatively small set of simple strategies and can therefore learn which ones work well.

There is a wealth of literature describing repetitive games and illustrating how future interactions provide incentives for collaboration. Much of the earlier literature suggests social control. The transition from public monitoring to private monitoring that corporate the differences in the observations of the players can dramatically complicate coordination and the provision of incentives, resulting in an equilibrium with private monitoring often seem unrealistically complex or fragile. Here's a model where players collaborate in an intuitively plausible way.

On the positive side, this model provides a simple and plausible theory of how players manage to collaborate despite the signals being private [1]. Strategic options

are limited, but enough to highlight (and isolate) two key differences between public and private monitoring games:

a) difficulty in providing incentives to start signal punishments: a player can ignore a bad signal by betting that he is the only one who received a bad signal.
b) difficulties in coordinating cooperation after the start of the punishment stage, as there is no longer a public signal that players could use to return to cooperation at the same time.

With regard to question (a), it is considered that the creation of incentives for the application of punishment requires significant costs for continued cooperation, while other shortcomings are significant. With regard to question (b), mental systems that generate forgiveness (i.e. respond to good signals) and some leniency (i.e. do not respond too strongly to bad signals) facilitate cooperation [3].

The second model to be considered in my research is the prisoner's dilemma with signals similar to the previous model of my research. In the prisoner's recurring dilemma [7], players make decisions repeatedly, knowing the previous outcomes of the game. I will consider the changed game, where each stage of each player will receive a signal about the decision that the opponent has already made.

In this model, the signals are private. The most important problem of the game is whether to believe the signals, and what is the probability that the signal is correct. Also, I should understand whether player should always believe the signals. Should we believe the signals, if the opponent does not cooperate several stages in a row. It is necessary to understand how many stages in a row it is necessary that opponent not to cooperate, so that the player stops listening to signals.

Chapter 1. First game-theoretic model

Gift exchange: There are two players who exchange gifts each period. Each of them has two possible actions, one of which corresponds to the lack of effort when choosing a gift, and the second-an expensive effort. Gifts may or may not be perceived as “thoughtful," and most likely a gift is perceived as thoughtful when the other is making an expensive effort.

Payoff structure: Actions are {C, D} with C presentation of costly efforts. The expected payouts for players are as follows:

	C	D
C	1, 1	-L, 1+L
D	1+L, -L	0, 0

where L corresponds to the cost of effort.

Signal structure: Suppose that there are two possible private signals that a player can receive, $y_i \in Y_i = \{0, 1\}$, where the signal correlates with the actions of another player.

Formally,

$$p = P\{y_i = 0 | a_j = D\} = P\{y_i = 1 | a_j = C\} \quad (1)$$

I assume that in (1) p >1/2 so that one can refer to $y_i = 0$ as a “bad” signal and $y_i = 1$ as “good”.

In addition to this private signal, I assume that at the beginning of each period players receive a public signal $z \in Z = \{0, 1\}$, and I let

$$q = P\{z = 1\} \quad (2)$$

1.1 Strategies

As mentioned above, the behavior of players in any period will depend on the previous game, but in a more limited way than in traditional models. The player is endowed with a mental system, which consists of a finite set of Si mental states in which the player can be, and the transition function T_i it describes what causes the movement from one state to another: function T_i determines the mental state of player i at the beginning of period t depending on his state in the period $t-1$, his choice of action in period $t-1$, and the results of this period and possibly previous periods. The restriction that is introduced is that the player can only condition his behavior by his mental state, not by the finer details of the story. Given this limitation, all comparisons between states and actions are considered acceptable. The set of pure strategies of the player i's is:

$$\Sigma_i = \{\sigma_i : S_i \longrightarrow A_i\}$$

I will illustrate the basic ideas with an example in which the players can be in one of two states U(pset) or N(ormal). The names of the two states are chosen in order to show that at any time player i is called to play an action, he knows the mood in which he is, which is a function of the story (own) of the game and the signals. Both S_i and T_i are exogenously given, not a choice: a player who made an effort in his choice of gift, did not receive a bad signal, may find it impossible not to get upset, that is to be in a U state.

The transition function for the example, which I will refer to as the leading example below, is as in Fig.1

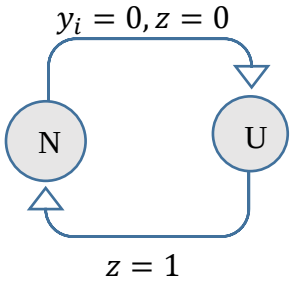


Fig.1. Transition.

Fig.1 shows which combinations of actions and signals lead to the transition from one state to another. If player i is in state N, he remains in this state until it receives signals $y_i = 0$ and $z = 0$, in which case he goes to state U. If i is in state U, he remains in this state until the signal is received $z = 0$, at this point, it enters the N state regardless of the signal y_i. A mental system that defines how observations are aggregated over time, hence how stories are combined: some stories lead to a state of N, others lead to state U.

The simple form of the transition function is that, given the previous state and the action taken, it depends only on the most recent signal – for simplicity. In principle, the transition function may depend on more than this last signal, for example, whether two of the last three signals were "bad" or they could also be stochastic.

Given the mental system above, our players behavior for each player i will be next,

σ_i(N) = C

σ_i(U) = D.

That is, player i plays C as long as $y_i = 1$ or when $z = 1$. He plays D otherwise. Player 1 causes a "punishment stage" when he sees a bad signal $y_i = 0$. This punishment stage ends only when signal $z = 1$.

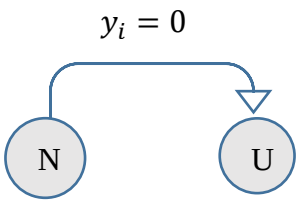


Fig.2. No "resetting".

The public signal z gives the possibility of "reset" relations in cooperative mode. If the signal z is ignored or absent and the thought process is determined by Fig.2, in the end, since the signals are noisy, with a probability of 1 players will get to the state U by the proposed strategy, and it will absorb: nothing will change in their behavior.

The signal z allows for possible recoordination back to state N (and possibly cooperation).

In example, players simultaneously switch from U to N for exogenous reasons. Alternatively, in a two-state mental system the players can move from state U back to state N after seeing a good signal.

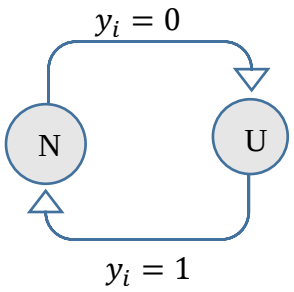


Fig.3. Forgiving transition.

Fig.3 describes the function of the transition. A player endowed with this alternative mental process, that will cooperate in N and defect in U, will follow the strategy a TIT for TAT strategy.

1.2 TIT for TAT strategy.

Tit for tat is a game theory mechanism subordinated to a payoff matrix similar to a prisoner's dilemma. Anatoly Rapoport [9], who developed a strategy in which each participant in the prisoner's recurring dilemma follows a course of action consistent with his opponent's previous move, presented the eye for an eye. For example, if a player is provoked, he responds with retribution; if not, he cooperates.

Tit for tat is a strategy that can be implemented in games with repeated moves or in a series of similar games. The concept revolves around game theory, an economic structure that explains how people interact with each other in a competitive environment. There are two types of game theory: cooperative game theory and non-cooperative game theory [7].

An eye for an eye claims that a person is more successful if he cooperates with another person. The implementation of the tit for tat strategy occurs when one agent collaborates with another agent in the very first interaction, and then simulates their next stages. This strategy is based on the concepts of retribution and altruism. Faced with a dilemma, an individual cooperates when another participant has a direct history of cooperation and does not fulfill obligations when the counterparty has not previously fulfilled obligations.

The most common example of this strategy is the prisoner's dilemma game. It will be discussed in Chapter 2 of this research.

1.3 Ergodic distributions and strategy evaluation

For any pair of strategies players will be ergodic distribution of pairs of played actions. The ergodic distribution gives the probability distribution of payoffs in a stage game, and I take the payoff to the players as the expected value of their payoffs given this distribution.

Formally define state profiles s as a pair of states (s_1, s_2,). Each strategy profile σ induces transition probabilities along the state profiles: under the assumption that each state profile s induces an action profile $\sigma(s)$which in turn generates a probability distribution over the signals and therefore, taking into account the transition functions T_i, over next states of the period. I denote by φ_σ the ergodic distribution over states induced by σ. That is, $\varphi_\sigma(s)$ corresponds to the (long-term) probability that players are in state s.

I associate with each strategy profile σ the value induced by the ergodic distribution. This corresponds to calculating the discounted expected payouts and accepting the discount to 1. Denote by $v(\sigma)$ this value (vector). Thus,

$$v(\sigma) = \sum_s g(\sigma(s))\phi_\sigma(s) \quad (3)$$

where $g(\sigma(s))$ is the payoff vector induced by the strategy profile σ for state profile s.

Equilibrium:

Definition. A profile $\sigma \in \Sigma$ is an equilibrium if for any player i and any strategy $\sigma_i' \in \Sigma_i$,

$v_{i(}\sigma_i', \sigma_{-i}) \leq v_i(\sigma)$.

This is a weaker notion of equilibrium than traditionally used in repeated games due to the limitation of the set of strategies to map from S_i to A_i.

1.4 Stages of cooperation and punishment

I consider next the ergodic distribution induced by our candidate equilibrium strategy σ for p close to 1 and examine in turn possible deviations from that strategy. The transition overstate profiles induced by σ is illustrated in figure 4.

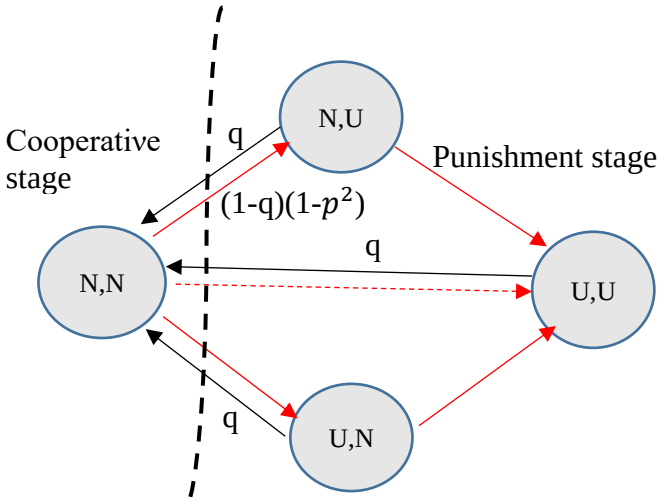


Figure 4: Transition over state profiles under σ.

When players follow our candidate equilibrium strategy, they alternate between cooperation and punishment stages. The probability of switching from cooperation to a punishment stage is $\pi = (1-q)(1-p^2)$ (since switching occurs when either player receives a bad signal and $z = 0$).The probability of switching from punishment to cooperation is q. Hence cooperative stages last on average $1/\pi$ periods, while punishment stages last on average $1/q$ periods. When player i plays D at both N and U, player j continues to alternate between stages of cooperation and defection. This is illustrated in figure 5.

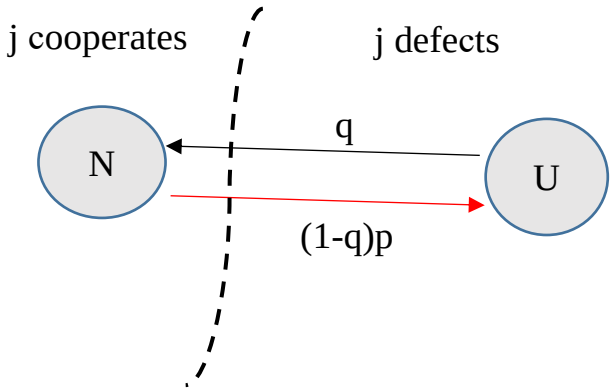


Figure 5: i defects always

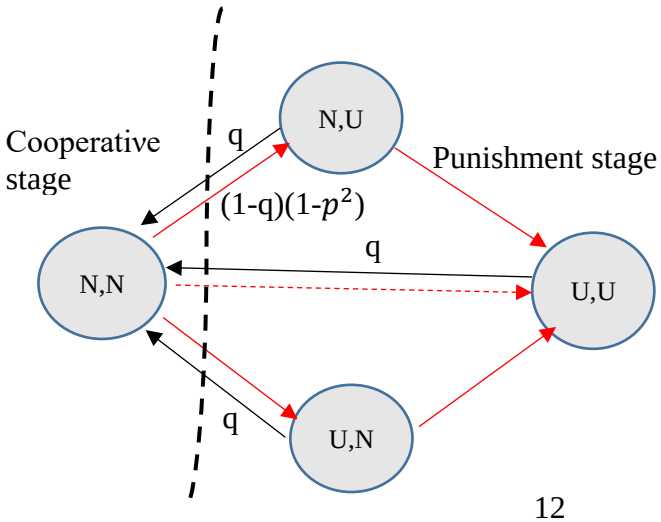
Player i receives a higher payoff in cooperation stages; however, those stages are now much shorter, as his opponent switches to state U with probability $(1-q)p$. For p close to 1,a defection at N almost certainly generates a bad signal, which start signals a punishment stage of expected length $1/q$ with probability $1-q$, hence an expected cost

$$\Delta = \frac{1-q}{q} \tag{6}$$

corresponding to a per-period decrease in payoff of 1 for an expected number of period equal to Δ in (6).

Deviation is deterred if $L < \Delta$ in (6).

When player i plays C at both N and U, he avoids start signaling some punishment stages, illustrated in figure 6.



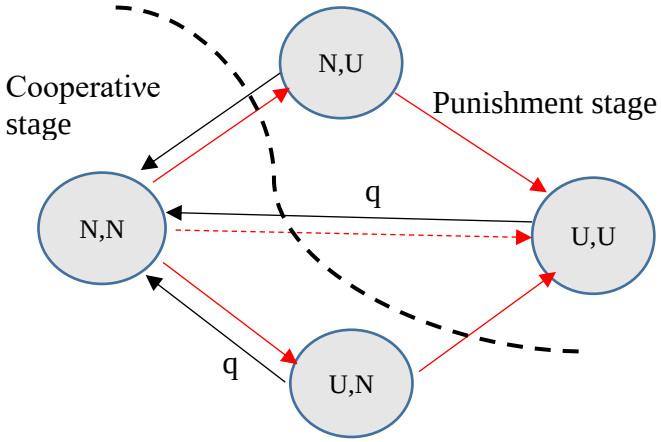


Figure 6: i cooperates always

However, he remains cooperative while player j is in a punishment stage. So L must be high enough for this option to be unattractive. More precisely, conditional on both players be in g in state N, there are events where only player i receives a bad signal, and events where only player j receives a bad signal. Under the first event, player i initially receives 1instead of $1 + L$, however he avoids the punishment stage, hence he makes a net gain of $\Delta - L$. Under the second event, nothing changes in the first period (because player i is still in state N), but he then receives $-L$ instead of 0 as long as the punishment stage continues, hence an expected cost equal to $L(1/q - 1) = L\Delta$. Since these two events have equal probability, playing C at N and U is not a profitable deviation if

$1/2(\Delta - L) + 1/2(-L\Delta) < 0$, that is

$$L > {}^{\Delta}/_{1+\Delta} \tag{7}$$

To summarize, for p close to1, there is a range of parameters (q, L) for which the proposed strategy is an equilibrium strategy. This range of parameters is such that ${}^{\Delta}/_{1+\Delta} < L < \Delta$, where $\Delta = {}^{1}/_{q-1}$. It is easy to check that outside this range, the only equilibrium entails both players defecting at both states. Similar analysis can be performed for other values of p.

1.5 Three mental states

The players in example had two possible mental states and two possible (pure) actions, which limited them to four pure strategies. This clearly limits them both in the existence of strategies that can lead to cooperation and in the possible profitable beneficial deviations. The addition of a state could allow for greater flexibility in responding to signals that could allow for cooperation that would not be possible only with two states. For example, when signals are not very informative and q is small, the prescribed strategies in the example may not be an equilibrium. When there is substantial chance that they got a bad signal when the other player chose C, I might prefer not to start signaling a punishment phase that can last for a long time. Thus, a combination of relatively inaccurate signals of enemy action and a low probability of avoiding a punishment regime can impede the balance in which they cooperate. The addition of a state could allow for cooperation that would be impossible only with two states in some respects. Assume that there is a state B in addition to the states N and U, and define transitions as follows:

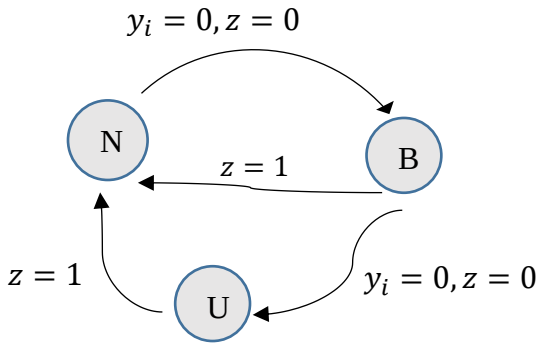


Figure 7: “Slow” transition to state U

The interpretation of this is that if a player in state N receives a bad signal, he goes to state B, not to the state U as in the example. If the first signal received a player in state B is bad, it goes to state U, and if it is positive, it returns to state N as if the bad signal that sent it to B has never been. Players remain in state U no matter what action is played or private signal received, until signal $z = 1$ occurs.

Signal $z = 1$ always sends players back to state N. The additional state, along with the transitional function amendments, allows the strategy to punish the opponent if there is some evidence that he is not cooperating, but the evidence that the punishment signals start can be either one bad signal (for example, the strategy plays C in N only, which I denote by $\bar{\sigma}$), or two consecutive bad signals (for example, the strategy plays C in N and B, and D in U, which I denote by η). Computations as above show that the strategy profile (η, η) that entails both players cooperating in states N and B and punishing in state U is an equilibrium for some parameters for which cooperation is impossible with two states. I have set the value q to 0.3. For the set of parameters (p, L) in the shaded area, (η, η) is an equilibrium profile. For the parameters (p, L) between the upper and lower curves (σ, σ) was an equilibrium profile in our simple two states.

Adding a state is not uniquely good. As mentioned above, the additional state allows not only more complex strategies to achieve cooperation, but also more complex strategies of rejection. Although both (σ, σ) and $(\bar{\sigma}, \bar{\sigma})$ generate the same behavior (under both profiles, as soon as the player observes a bad signal, he continues the defect until the signal $z = 1$ occurs), there are parameters for which (σ, σ) is an equilibrium, but $(\bar{\sigma}, \bar{\sigma})$ is not. For $(\bar{\sigma}, \bar{\sigma})$ to be an equilibrium, players must have an incentive to play D in M. This entails a strictly tighter restriction because playing C in M is cheaper than playing C at U: in case the opponent is already in U, the player will probably get a negative signal and switch from M to U the next period. For example, for p close to 1 the cost of playing C in M will be only one period, not corresponding to the duration of the punishment stage. Formally, the stimulating restriction becomes: $^1/_2\,(\Delta - L) + {}^1/_2\,(1-q)(-L) \geq 0$.

Rearranging the terms have $L \geq \frac{\Delta}{2-q} = \frac{1-q}{q(2-q)}$,

a condition that is always more restrictive than $L \geq \frac{\Delta}{1+\Delta} = 1 - q$

Chapter 2. Second game-theoretic model

The original version of the prisoner's dilemma game dealt with a situation in which two prisoners — accomplices to a crime — were interrogated in separate rooms. Each of the prisoners has a choice: either confess to the crime and thereby involve the other, or deny their participation in the crime. If only one of the prisoners confessed, he would be released and the charge would fall on the other prisoner, who would be sentenced to B months in prison. If both prisoners deny their involvement in the crime, both will be held in prison for C month due to formalities, and if both players confess, both will be sentenced to A months in prison. The entries in each cell of the matrix represent the usefulness attributed by each player to different prison terms, which we will, for simplicity, consider the duration of their imprisonment.

Put yourself in the shoes of player 1. If player 2 decides to deny having committed a crime, then of course you'd better confess, as then you'll be released. Similarly, if player 2 admits, then you'd better confess, as in this case you will not be sentenced to B months in prison, but only to A. Therefore, whatever player 2 does, it is more profitable for player 1 to confess.

Prisoner's dilemma: Two players sit in different cameras. Each period, each of them has two possible actions, one of which corresponds to confess to the crime, and the second - to remain silent.

Payoff structure: Actions are {C, D} with C is keep silence. The expected payoffs to the players are as follows:

	C	D
C	A, A	B, 0
D	0, B	C, C

Signal structure: Every period each player receives a signal about how his opponent will act in this stage. I am guessing there are two possible private signals that

player i might receive, $y_i \in Y_i = \{0, 1\}$, where a signal is correlated with the other player's action. Formally,

$$p = P\{y_i = 0 | a_j = D\} = P\{y_i = 1 | a_j = C\} \quad (8)$$

I am guessing that p >1/2 so that one can refer to $y_i = 0$ as a "bad" signal and $y_i = 1$ as "good".

2.1 Strategies

The behavior of players in any period will depend on the previous game, but in a more limited way than in traditional models. The player is endowed with a mental system, which consists of a finite set of Si mental states in which the player can be, and the transition function T_i it describes what causes the movement from one state to another: the function T_i determines the mental state of player i at the beginning of period t depending on his state in the period $t-1$, his choice of action in period $t-1$, and the results of this period and possibly previous periods. The restriction that is introduced is that the player can only condition his behavior by his mental state, not by the finer details of the story. Given this limitation, all comparisons between states and actions are considered acceptable. The set of pure strategies of the player i's is:

$$\Sigma_i = \{\sigma_i : S_i \longrightarrow A_i\}$$

As in the previous game, players can be in one of two states U(pset) or N(normal). In example, players simultaneously move from U to N, in a two states of the mental system, players can move from the state of U back to state N after seeing a good signal.

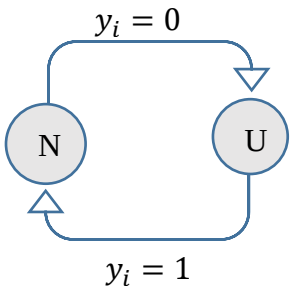


Fig.8. Forgiving transition.

Fig.8 corresponds to fig.3. in first game. It describes such a transition function, which was not suitable for previous game.

However, in this game players endowed with mental process, who would cooperate in N and defect in U, would be following a TIT for TAT strategy.

The simple form of the transition function is that, given the previous state and the action taken, it depends only on the most recent signal – for simplicity. In principle, the

transition function may depend on more than this last signal, for example, whether two of the last three signals were “bad” or they could also be stochastic.

Given the mental system above, our players behavior for each player i will be as follows,

σ_i(N) = C

σ_i(U) = D.

TIT for TAT strategy has been described in Chapter 1.2.

2.2 Successful cooperation

I am interested in balances in which the players cooperate at least some of the time asymptotically. This requires players playing the strategy “play C in N and D in U”.

Our question is, what is the probability that the signals (y_1, y_2) is correct, the players have to cooperate.

	C	D	
C	A, A	B, 0	p
D	0, B	C, C	1-p
	q	1-q	

Probabilities:

The probability that the first player will choose a strategy C: $\mu_1(C) = p$;

The probability that the first player will choose a strategy D: $\mu_1(D) = 1 - p$;

The probability that the second player will choose a strategy C: $\mu_2(C) = q$;

The probability that the second player will choose a strategy D: $\mu_2(D) = 1 - q$.

Let's write down the average payoff of the 1st player from the use of the 1st strategy:

$$\overline{\mu_1}(C) = Aq + B(1 - q) = q(A - B) + B$$

Similarly, the 2-nd strategy:

$$\overline{\mu_1}(D) = Bq + C(1 - q) = q(B - C) + C$$

Now let's compare these payoffs with each other:

$\overline{\mu_1}(C) > \overline{\mu_1}(D)$: $p = 1;\ q(A - B) + B > q(B - C) + C \implies q < \frac{C-B}{(A-2B+C)}$;

$\overline{\mu_1}(C) < \overline{\mu_1}(D)$: $p = 0;\ q(A - B) + B < q(B - C) + C \implies q > \frac{C-B}{(A-2B+C)}$;

$\overline{\mu_1}(C) = \overline{\mu_1}(D)$: $p \in [0,1];\ q(A - B) + B = q(B - C) + C \implies q = \frac{C-B}{(A-2B+C)}$.

The first player's response (best response) to the second player's actions:

$$R_1(q) \equiv p = \begin{cases} 1, q < \dfrac{C-B}{(A-2B+C)} \\ 0, q > \dfrac{C-B}{(A-2B+C)} \\ [0,1], q = \dfrac{C-B}{(A-2B+C)} \end{cases}$$

Let's write down the average payoff of the 2nd player from using his 1st strategy:

$$\overline{\mu_2}(C) = Ap + B(1-p) = p(A-B) + B$$

Similarly, the 2-nd strategy:

$$\overline{\mu_2}(D) = Bp + C(1-p) = p(B-C) + C$$

The second player's response function (best response) to the actions of the first player is similar:

$$R_2(p) \equiv q = \begin{cases} 1, p < \dfrac{C-B}{(A-2B+C)} \\ 0, p > \dfrac{C-B}{(A-2B+C)} \\ [0,1], p = \dfrac{C-B}{(A-2B+C)} \end{cases}$$

For players the probabilities that opponent will choose a strategy C is $p < \frac{C-B}{(A-2B+C)}$ for 1'st player and for 2'nd player is $q < \frac{C-B}{(A-2B+C)}$.

So the player suspects that his opponent can deviate with such probability: $p = q > \frac{C-B}{(A-2B+C)}$. Thus, the player has his own intuitive (private) signal that his opponent will deviate from cooperation. And if this signal $p = q > \frac{C-B}{(A-2B+C)}$, the player does not cooperate.

In that way, each stage of the game, to each player comes his private signal:

$$y_i \in Y_i = \{0, 1\}.$$

For the 1'st player: $y_1 \in Y_1 = \{0, 1\}$;

For the 2'nd player: $y_2 \in Y_2 = \{0, 1\}$.

And each stage of the game, the player chooses whether to cooperate with his opponent, if $y_i < \frac{C-B}{(A-2B+C)}$.

2.3 Non-cooperation many stages in a row

In this game, players listen to signals, but is it reasonable to listen to signals if the opponent does not cooperate several stages in a row. It is necessary to understand how many stages in a row the opponent's refusal to cooperate is crucial for the player to stop listening to the signals.

Firstly, need to calculate at what stage doesn't make sense to punish the opponent who refused cooperation.

Suppose that our game has n stages.

Our players cooperate $k-1$ stages and on k stage, one of players non-cooperate.

$k-1$ k stage

Coop … Coop Non-coop …

Coop … Coop Coop …

Player 2 can select punishment phase, but he should understand, it will help to make the opponent payoffs less than with cooperative phases:

$$A(k-1)+Bk+C(n-k)<An$$

$$k(A+B-C)<A-n(C-A)$$

$$k<\frac{A-n(C-A)}{(A+B-C)}$$

If a stage k where the opponent has ceased to cooperate is: $<\frac{A-n(C-A)}{(A+B-C)}$, player can select punishment phase.

In our example, players listen signals. But if opponent are non-cooperate, how long player should listen signal. It is necessary to count how many stages in a row the refusal of cooperation of the opponent is acceptable that the player listened to a signal.

$k-1$ m stages

Coop … Coop Non-coop … Non-coop …

Coop … Coop Coop … Coop …

$$A(k-1)+Bm+C(n-k-m+1)<An$$

$$k(A-C)+m(B-C)+n(C-A)<(A-C)$$

$$m(B-C)<(A-C)-n(C-A)-k(A-C)$$

$$m<\frac{(A-C)(1+n-k)}{(B-C)}$$

If the opponent does not cooperate more than $m>\frac{(A-C)(1+n-k)}{(B-C)}$ stages in a row, then it makes no sense to listen to the signals, it is necessary to move into the phase of punishment.

2.4 The prisoner's Dilemma game, example

$G(t,T)$ - game $G(T)$, arising after $t-1$ repetitions of the game G.

Example. The prisoner's Dilemma game:

	C	D
C	7, 7	10, 0
D	0, 10	5, 5

Probabilities:

The probability that the first player will choose a strategy C: $\mu_1(C) = p$;

The probability that the first player will choose a strategy D: $\mu_1(D) = 1 - p$;

The probability that the second player will choose a strategy C: $\mu_2(C) = q$;

The probability that the second player will choose a strategy D: $\mu_2(D) = 1 - q$.

Let's write down the average payoff of the 1st player from the use of the 1st strategy:

$$\overline{\mu_1}(C) = 7q + 10(1-q) = (-3)q + 10$$

Similarly, the 2-nd strategy:

$$\overline{\mu_1}(D) = 10q + 5(1-q) = 5q + 5$$

Now let's compare these payoffs with each other:

$\overline{\mu_1}(C) > \overline{\mu_1}(D)$: $p = 1;\ (-3)q + 10 > 5q + 5 \Longrightarrow q < \frac{5}{8}$;

$\overline{\mu_1}(C) < \overline{\mu_1}(D)$: $p = 0;\ (-3)q + 10 < 5q + 5 \Longrightarrow q > \frac{5}{8}$;

$\overline{\mu_1}(C) = \overline{\mu_1}(D)$: $p \in [0,1];\ (-3)q + 10 = 5q + 5 \Longrightarrow q = \frac{5}{8}$.

The first player's response (best response) to the second player's actions:

$$R_1(q) \equiv p = \begin{cases} 1, q < \frac{5}{8} \\ 0, q > \frac{5}{8} \\ [0,1], q = \frac{5}{8} \end{cases}$$

Let's write down the average payoff of the 2nd player from using his 1st strategy:

$$\overline{\mu_2}(C) = 7q + 10(1-q) = (-3)q + 10$$

Similarly, the 2-nd strategy:

$$\overline{\mu_2}(D) = 10q + 5(1-q) = 5q + 5$$

The second player's response function (best response) to the actions of the first player is similar:

$$R_2(p) \equiv q = \begin{cases} 1, q < \frac{5}{8} \\ 0, q > \frac{5}{8} \\ [0,1], q = \frac{5}{8} \end{cases}$$

Based on the calculations, for players the probabilities that opponent will choose a strategy C is $p < \frac{5}{8}$ for 1'st player and for 2'nd player is $q < \frac{5}{8}$.

So the player suspects that his opponent can deviate with such probability: $= q > \frac{5}{8}$. Thus, the player has his own intuitive (private) signal that his opponent will deviate from cooperation. And if this signal $= q > \frac{5}{8}$, the player does not cooperate.

In that way, each stage of the game, to each player comes his private signal:

$$y_i \in Y_i = \{0, 1\}.$$

For the 1'st player: $y_1 \in Y_1 = \{0, 1\}$;

For the 2'nd player: $y_2 \in Y_2 = \{0, 1\}$.

And each stage of the game, the player chooses whether to cooperate with his opponent, if $y_i < \frac{5}{8}$.

In this example 10 stages.

Our players cooperate $k - 1$ stages and on k stage, one of players non-cooperate.

$k-1$ k stage

Coop … Coop Non-coop …

Coop … Coop Coop …

Player 2 can select punishment phase, but he should understand, it will help to make the opponent payoffs less than with cooperative phases:

$$7(k-1)+10k+5(10-k)<10*7$$

$$k<\frac{7-10(5-7)}{(7+10-5)}$$

$$k<\frac{27}{12}\cong 2{,}25$$

If opponent start non-cooperate less than on 2'nd stage ($k=2{,}25$), player can select punishment phase.

$k-1$ m stages

Coop … Coop Non-coop … Non-coop …

Coop … Coop Coop … Coop …

$$7(2{,}25-1)+10m+5(10-2{,}25-m+1)<7*10$$

$$m<\frac{(7-5)(1+10-2{,}25)}{(10-5)}$$

$$m<\frac{17{,}5}{5}\cong 3{,}5$$

If the opponent does not cooperate more than $m>3{,}5$ stages in a row, then it makes no sense to listen to the signals, it is necessary to move into the phase of punishment.

Chapter 3. Software implementation

3.1 Software implementation of first game

The c# programming language and Nisual Studio programming environment were chosen to write the program.

At the first stage, the program asks you to enter the specified signal probability $z = 1$ and the number of stages of the game:

```
Console.WriteLine("Probability z=1 is and number of stages");
            string UserInput = Console.ReadLine();
            List<float> inputs = ParseInput(UserInput);
            uint stageNalue;

            bool stagesNalueCorrect = false;
            if(inputs.Count >= 2)
            {
                stagesNalueCorrect = uint.TryParse(inputs[1].ToString(),
out stageNalue);
            }
            while (!(inputs.Count >= 2 && stagesNalueCorrect && inputs[0]
>= 0 && inputs[0] <= 1))
```

If the data format is incorrect, the program will inform you about it:

```
                Console.WriteLine("Wrong format! Probability z=1 is and
number of stages");
                UserInput = Console.ReadLine();
                inputs = ParseInput(UserInput);
                if (inputs.Count >= 2)
                {
                    stagesNalueCorrect =
uint.TryParse(inputs[1].ToString(), out stageNalue);
                }
            }
            stagesNalueCorrect = uint.TryParse(inputs[1].ToString(), out
stageNalue);
            if (stagesNalueCorrect)
            {
                Console.WriteLine("Entered numbers: q = {0}, S = {1} ",
inputs[0], stageNalue);
```

Next, the program offers to enter private signals at stage 0 for players. For the next stages, the program generates them itself:

```
Console.WriteLine("\r\nEnter the first space-separated signals (y1
=(0,1); y2 =(0,1)) received by player 1 and player 2");
            string UserInputStrategies = Console.ReadLine();
            List<char> inputsStrategies =
ParseInputStrategies(UserInputStrategies);
            while(inputsStrategies.Count < 2)
```

If the data format is incorrect, the program will inform you about it:

```
Console.WriteLine("Wrong format! Enter the first space-separated signals
(y1 =(0,1); y2 =(0,1)) received by player 1 and player 2");
            UserInputStrategies = Console.ReadLine();
            inputsStrategies = ParseInputStrategies(UserInputStrategies);
```

In charter 1.4 and charter 1.5 restrictions were found for *L*:

$$\Delta/_{1+\Delta} < L < \Delta, \text{ where } \Delta = 1/_{q-1}.$$

I should put them on the program:

```
Random rd = new Random();
         float q = inputs[0];
         int leftSide = (int)Math.Round((1 - q) * 1000);
         int rightSide = (int)Math.Round((1 - q) * 1000 / q);
         float L = Convert.ToSingle(rd.Next(leftSide, rightSide)) / 1000;
```

Here is the calculation of L and the probability with which players believe private signals is indicated:

```
while (stage < stageNalue)
            {
               L = Convert.ToSingle(rd.Next(leftSide, rightSide)) / 1000;

                int threshold = 30;

               r1 = rd.Next(0, 101) > threshold ? 1 : 0;
               r2 = rd.Next(0, 101) > threshold ? 1 : 0;

                z = rd.Next(0, 2);
```

The model of the game was indicated in Charter 1.

Payoff structure: Actions are {C, D} with C representing costly effort. The expected payoffs to the players are as follows:

	C	D
C	1, 1	-L, 1+L
D	1+L, -L	0, 0

where L corresponds to the cost of effort.

Introduce the structure of winnings:

```
static string GetMatrixResult(char CurrentPlayer1Choice, char
CurrentPlayer2Choice, float L, float[] result)
        {
            string matrixResult = "";
            if (CurrentPlayer1Choice == '1')
            {
                if (CurrentPlayer2Choice == '1')
                {
                    matrixResult = "1,1";
                    result[0] += 1;
                    result[1] += 1;
                }
                else
                {
                    matrixResult = (-L).ToString() + "; " + (1 +
L).ToString();
                    result[0] += -L;
                    result[1] += 1+L;
                }
            }
            else
            {
                if (CurrentPlayer2Choice == '1')
                {
                    matrixResult = (1 + L).ToString() + "; " + (-
L).ToString();
                    result[0] += 1+L;
                    result[1] += -L;
                }
                else
                {
                    matrixResult = "0,0";
                    result[0] += 0;
                    result[1] += 0;
                }
            }

            return matrixResult;
      }
```

Thus, the number of game stages, the probability of the signal z=1, and private signals at stage 0 are fed to the program input. The output receives data signals at each stage and the winnings of the players.

Check the operation of the program on several examples.

Example 1.

In charter 1 it was considered that $q = 0.3$. Let's use this as an example.

Let's check the model for a small number of stages - 5.

```
Probability z=1 is and number of stages
0,3 5
Entered numbers: q = 0,3, S = 5
```

The first private signals assume good ($y_1 = 1, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
1 1
Entered signals: 1 = 1, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = C, Player2 = C
y1 = 1, y2 = 1
1,1


Stage 1: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 1
1,1


Stage 2: Player1 = D, Player2 = D.
y1 = 0, y2 = 0, z = 0
0,0


Stage 3: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


Stage 4: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 1
1,1
```

```
Stage 5: Player1 = C, Player2 = D.
y1 = 1, y2 = 0, z = 0
-2,084; 3,084


The outcome of the games: 1,916; 7,084
```

At each stage, you can see the strategies that the players have chosen, the signals received and the winnings. At the end, the players ' winnings for the whole game are calculated.

Example 2.

Let's use this as an example $q = 0.5$.

Let's check the model for a 10 number of stages.

```
Probability z=1 is and number of stages
0,5 10
Entered numbers: q = 0,5, S = 10
```

The first private signals assume good ($y_1 = 1, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
1 1
Entered signals: 1 = 1, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = C, Player2 = C
y1 = 1, y2 = 1
1,1


Stage 1: Player1 = C, Player2 = D.
y1 = 1, y2 = 0, z = 0
-0,523; 1,523


Stage 2: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1
```

```
Stage 3: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 0
1,1


Stage 4: Player1 = C, Player2 = D.
y1 = 1, y2 = 0, z = 0
-0,63; 1,63


Stage 5: Player1 = D, Player2 = D.
y1 = 0, y2 = 0, z = 0
0,0


Stage 6: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


Stage 7: Player1 = D, Player2 = D.
y1 = 0, y2 = 0, z = 0
0,0


Stage 8: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


Stage 9: Player1 = D, Player2 = D.
y1 = 0, y2 = 0, z = 0
0,0


Stage 10: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


The outcome of the games: 4,847; 9,153
```

Example 3.

Let's use this as an example $q = 0.1$.

Let's check the model for a 8 number of stages.

```
Probability z=1 is and number of stages
0,1 8
Entered numbers: q = 0,1, S = 8
```

The first private signal for player 0 assume good and for player 1 assume bad ($y_1 = 0, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
0 1
Entered signals: 1 = 0, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = D, Player2 = C
y1 = 0, y2 = 1
4,726; -3,726


Stage 1: Player1 = D, Player2 = D.
y1 = 0, y2 = 0, z = 0
0,0


Stage 2: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


Stage 3: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 1
1,1


Stage 4: Player1 = C, Player2 = D.
y1 = 1, y2 = 0, z = 0
-1,419; 2,419


Stage 5: Player1 = C, Player2 = C.
y1 = 0, y2 = 0, z = 1
1,1


Stage 6: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 0
1,1
```

Stage 7: Player1 = C, Player2 = C.
y1 = 1, y2 = 1, z = 0
1,1

Stage 8: Player1 = C, Player2 = C.
y1 = 1, y2 = 0, z = 1
1,1

The outcome of the games: 9,306999; 4,693

3.2 Software implementation of second game

At the first stage, the program asks you to enter the number of stages of the game:

```
Console.WriteLine("Number of stages ");
string UserInput = Console.ReadLine();
uint stages = ParseInput(UserInput);

while (stages == 0)
{
    Console.WriteLine("Wrong format! Number of stages ");
    UserInput = Console.ReadLine();
    stages = ParseInput(UserInput);
}
if (stages != 0)
{
    Console.WriteLine("number of stages: S = {0} ", stages);
}
#endregion
```

Next, the program offers to enter payoffs for matrix (A, B, C):

```
Console.WriteLine("\r\n Enter A, B, C for the matrix: \r\n AA B0 \r\n 0B
CC");
            string UserInputMatrix = Console.ReadLine();
            List<float> inputsMatrix = ParseInputMatrix(UserInputMatrix);
            while (inputsMatrix.Count < 3)
            {
                Console.WriteLine("Wrong format! Enter A, B, C for the
matrix: \r\n AA B0 \r\n 0B CC");
                UserInputMatrix = Console.ReadLine();
                inputsMatrix = ParseInputMatrix(UserInputMatrix);
            }
            Console.WriteLine("Entered matrix: \r\n {0},{0}   {1},0 \r\n
0,{1}   {2},{2}", inputsMatrix[0], inputsMatrix[1], inputsMatrix[2]);
            #endregion
```

Next, the program offers to enter private signals at stage 0 for players. For the next stages, the program generates them itself:

```
Console.WriteLine("\r\nEnter the first space-separated signals (y1
=(0,1); y2 =(0,1)) received by player 1 and player 2");
            string UserInputStrategies = Console.ReadLine();
            List<char> inputsStrategies =
ParseInputStrategies(UserInputStrategies);
            while(inputsStrategies.Count < 2)
```

If the data format is incorrect, the program will inform you about it:

```
Console.WriteLine("Wrong format! Enter the first space-separated signals
(y1 =(0,1); y2 =(0,1)) received by player 1 and player 2");
            UserInputStrategies = Console.ReadLine();
            inputsStrategies = ParseInputStrategies(UserInputStrategies);
```

In charter 2.2 and charter 2.3 restrictions were found for p:

$$p < \frac{C-B}{(A-2B+C)}.$$

I should put them on the program:

```
float A = inputsMatrix[0];
            float B = inputsMatrix[1];
            float C = inputsMatrix[2];
            float r = (C-B) / (A -2*B + C);
```

The model of the game was indicated in Charter 2.

Payoff structure: Actions are {C, D} with C representing costly effort. The expected payoffs to the players are as follows:

	C	D
C	A, A	B, 0
D	0, B	C, C

Introduce the structure of winnings:

```
static string GetMatrixResult(char CurrentPlayer1Choice, char
CurrentPlayer2Choice, float[] result)
        {
            string matrixResult = "";
            if (CurrentPlayer1Choice == '1')
            {
                if (CurrentPlayer2Choice == '1')
                {
                    matrixResult = "1,1";
                    result[0] += A;
                    result[1] += A;
                }
                else
                {
                   matrixResult = (B).ToString() + "; " + (0).ToString();
                   result[0] += B;
```

```
                result[1] += 0;
            }
        }
        else
        {
            if (CurrentPlayer2Choice == '1')
            {
                matrixResult = (B).ToString() + "; " + (0).ToString();
                result[0] += B;
                result[1] += 0;
            }
            else
            {
                matrixResult = "C,C";
                result[0] += C;
                result[1] += C;
            }
        }

        return matrixResult;
    }
```

Thus, the number of game stages and private signals at stage 0 are fed to the program input. The output is the signal data at each stage and the winnings of the players.

Check the operation of the program on several examples.

Example 1.

In charter 2.3 it was considered that $A = 7, B = 10, C = 5$. Let's use this as an example.

Let's check the model for a small number of stages - 5.

```
Number of stages
10
Number of stages: S = 10

Enter the payoffs A, B, C for matrix:
 AA B0
 0B CC
7 10 5
Matrix:
 7,7   10,0
 0,10   5,5
```

The first private signals assume good ($y_1 = 1, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
1 1
Entered signals: 1 = 1, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = C, Player2 = C
y1 = 1, y2 = 1
7,7


Stage 1: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
7,7


Stage 2: Player1 = D, Player2 = D.
y1 = 0, y2 = 0
5,5


Stage 3: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
7,7


Stage 4: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
7,7


Stage 5: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
10; 0


Stage 6: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
10; 0


Stage 7: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
10; 0
```

```
Stage 5: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
10; 0


Stage 5: Player1 = D, Player2 = D.
y1 = 1, y2 = 0
5; 5


Stage 5: Player1 = D, Player2 = D.
y1 = 1, y2 = 0
5; 5


The outcome of the games: 83; 43
```

Example 2.

Let's use this as an example $A = 3, B = 5, C = 2$.

Let's check the model for a 10 number of stages.

```
Number of stages
10
Number of stages: S = 10

Enter the payoffs A, B, C for matrix:
 AA B0
 0B CC
3 5 2
Matrix:
 3,3   5,0
 0,5   2,2
```

The first private signals assume good ($y_1 = 1, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
1 1
Entered signals: 1 = 1, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = C, Player2 = C
y1 = 1, y2 = 1
3,3
```

```
Stage 1: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
5; 0


Stage 2: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
3,3


Stage 3: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
3,3


Stage 4: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
5; 0


Stage 5: Player1 = D, Player2 = D.
y1 = 0, y2 = 0
2,2


Stage 6: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
3,3


Stage 7: Player1 = D, Player2 = D.
y1 = 0, y2 = 0
2,2


Stage 8: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
3,3


Stage 9: Player1 = D, Player2 = D.
y1 = 0, y2 = 0
2,2


Stage 10: Player1 = C, Player2 = C.
```

```
y1 = 0, y2 = 0
3,3


The outcome of the games: 34; 24
```

Example 3.

Let's use this as an example $A = 5, B = 20, C = 10$.

Let's check the model for a 8 number of stages.

```
Number of stages
8
Number of stages: S = 8

Enter the payoffs A, B, C for matrix:
 AA B0
 0B CC
5 20 10
Matrix:
 5,5   20,0
 0,20   10,10
```

The first private signal for player 0 assume good and for player 1 assume bad ($y_1 = 0, y_2 = 1$):

```
Enter the first space-separated signals (y1 =(0,1); y2 =(0,1)) received
by player 1 and player 2
0 1
Entered signals: 1 = 0, 2 = 1
```

The result of the program:

```
Stage 0: Player1 = D, Player2 = C
y1 = 0, y2 = 1
0; 20


Stage 1: Player1 = D, Player2 = D.
y1 = 0, y2 = 0
10,10


Stage 2: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
5,5
```

```
Stage 3: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
5,5


Stage 4: Player1 = C, Player2 = D.
y1 = 1, y2 = 0
20;0


Stage 5: Player1 = C, Player2 = C.
y1 = 0, y2 = 0
5,5


Stage 6: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
5,5


Stage 7: Player1 = C, Player2 = C.
y1 = 1, y2 = 1
5,5


Stage 8: Player1 = C, Player2 = C.
y1 = 1, y2 = 0
20,0


The outcome of the games: 75; 35
```

Conclusion

The aim of this research was to study the impact of the mental system of players and receive signals from the outside, on the possibility of cooperation. The goal was achieved.

The objectives of this research were to study the literature on this problem, and write software products that illustrate the behavior of players for the games studied.

As a result of this research, two games of game theory were considered.

In the first model, the game that two players give each other gifts was considered. The game takes place with the participation of signals with two mental systems of players. Restrictions on L in the matrix were found. After behavior of players with three mental systems was considered and also restrictions on L in the matrix were found.

In the second model, the prisoners' dilemma is considered, but also with the participation of signals (similar to the first game). In the general case was found - what is the probability that the signals (y_1, y_2) is correct, the players have to cooperate. One numerical example of the prisoner's dilemma was solved.

Next, for each of the games were created software algorithms in C# in Nisual Studio. Software algorithms automatically calculate all calculations performed for existing games. Software algorithms were successfully tested on new and existing examples.

The studied problems in the research are relevant, because the problem with the introduction of mental systems and extraneous signals - it is still a little studied problems. These problems are only gaining popularity.

List of literature

1. Compte, O., Postlewaite, A. 2015. Plausible cooperation. Mimeo, University of Pennsylvania.
2. Compte, O., Postlewaite, A. 2012. Strategy restrictions and limited knowledge. Mimeo, University of Pennsylvania.
3. Compte, O., Postlewaite, A. 2013. Belief free balances. Mimeo, University of Pennsylvania.
4. Hauser, C., Hopenhayn, H., 2008. Trading favors: optimal exchange and forgiveness. Mimeo, UCLA.
5. Kandori, M. and H. Matsushima 1998, "Private Observation, Communication and Collu-sion."Econometrica, Nol. 66, 627-652.
6. Monte, D., 2007. Reputation and bounded memory: a theory of inertia and skepticism. Mimeo.
7. Neumann J., Morgenstern O., 1947. Theory of games and economic behavior. Princeton: Princeton Univ.
8. Piccione, M. 2002, The repeated prisoner's dilemma with imperfect private monitoring, Journal of Economic Theory102 , 70—83.
9. Romero, J., 2010. Bounded rationality in repeated games. Mimeo, Caltech.
10. Sainty, Barbara 1999, "Achieving Greater Cooperation in a Noisy Prisoner's Dilemma: An Experimental Investigation." Journal of Economic Behavior and Organization, 39(4), pp.421—35.
11. Wilson, A. 2004, "Bounded Memory and Biases in Information Processing, University of Chicago.

Appendix

Appendix 1. Program implementation of the first model

```
using System;
using System.Collections.Generic;

namespace Alexandra
{
    class Program
    {
        static void Main(string[] args)
        {

            #region Считывание q и S
            Console.WriteLine("Probability z=1 is and number of stages");
            string UserInput = Console.ReadLine();
            List<float> inputs = ParseInput(UserInput);
            uint stageNalue;

            bool stagesNalueCorrect = false;
            if(inputs.Count >= 2)
            {
                stagesNalueCorrect = uint.TryParse(inputs[1].ToString(), out stageNalue);
            }
            while (!(inputs.Count >= 2 && stagesNalueCorrect && inputs[0] >= 0 && inputs[0]
<= 1))
            {
                Console.WriteLine("Wrong format! Probability z=1 is and number of stages");
                UserInput = Console.ReadLine();
                inputs = ParseInput(UserInput);
                if (inputs.Count >= 2)
                {
                    stagesNalueCorrect = uint.TryParse(inputs[1].ToString(), out
stageNalue);
                }
            }
            stagesNalueCorrect = uint.TryParse(inputs[1].ToString(), out stageNalue);
            if (stagesNalueCorrect)
            {
                Console.WriteLine("Entered numbers: q = {0}, S = {1} ", inputs[0],
stageNalue);
            }
            #endregion

            #region Считывание усилий Playerов
            Console.WriteLine("\r\nEnter the first space-separated signals (y1 =(0,1); y2
=(0,1)) received by player 1 and player 2");
            string UserInputStrategies = Console.ReadLine();
            List<char> inputsStrategies = ParseInputStrategies(UserInputStrategies);
            while(inputsStrategies.Count < 2)
            {
                Console.WriteLine("Wrong format! Enter the first space-separated signals (y1
=(0,1); y2 =(0,1)) received by player 1 and player 2");
                UserInputStrategies = Console.ReadLine();
                inputsStrategies = ParseInputStrategies(UserInputStrategies);
            }
```

```
            Console.WriteLine("Entered signals: 1 = {0}, 2 = {1} ", inputsStrategies[0],
inputsStrategies[1]);
            #endregion

            #region Инициализация L
            Random rd = new Random();
            float q = inputs[0];
            int leftSide = (int)Math.Round((1 - q) * 1000);
            int rightSide = (int)Math.Round((1 - q) * 1000 / q);
            float L = Convert.ToSingle(rd.Next(leftSide, rightSide)) / 1000;
            #endregion


            #region Инициализация 1 Stagea
            float[] result = new float[2] { 0, 0 };

            int stage = 0;
            int r1, r2, z;


            char CurrentPlayer1Choice = inputsStrategies[0];
            char CurrentPlayer2Choice = inputsStrategies[1];

            string matrixResult = "";

            //Первый Stage
            char y1 = CurrentPlayer1Choice;
            char y2 = CurrentPlayer2Choice;

            matrixResult = GetMatrixResult(CurrentPlayer1Choice, CurrentPlayer2Choice, L,
result);

            Console.WriteLine("\r\n\r\nStage {0}: Player1 = {1}, Player2 = {2}", stage,
CurrentPlayer1Choice, CurrentPlayer2Choice);
            Console.WriteLine("y1 = {0}, y2 = {1}", y1, y2);
            Console.WriteLine(matrixResult);
            #endregion

            #region 2 и следующие Stageи
            //ВТОРОЙ И СЛЕДУЮЩИЙ STAGEИ
            while (stage < stageNalue)
            {
                L = Convert.ToSingle(rd.Next(leftSide, rightSide)) / 1000;

                int threshold = 30; //30% - вероятность, что примет 0, 70% - вероятность,
что примет 1. Значение threshold располагается в диапазоне 0-100.

                r1 = rd.Next(0, 101) > threshold ? 1 : 0;
                r2 = rd.Next(0, 101) > threshold ? 1 : 0;

                z = rd.Next(0, 2);


                //ДЛЯ PLAYERA 1 НА ТЕКУЩЕМ STAGEE
                if(CurrentPlayer1Choice == '1')
                {
                    if(r1 == 1)
                    {
                        y1 = CurrentPlayer2Choice;
                    }
```

```
                    else //r1 = 0
                    {
                        y1 = CurrentPlayer2Choice == '1' ? '0' : '1';               //Другой
Player на предыдущем Stagee НАОБОРОТ
                    }
                    CurrentPlayer1Choice = (z == 0 && y1 == '0') ? '0' : '1';
                }
                else
                {
                    CurrentPlayer1Choice = (z == 1) ? '1' : '0';
                }


                //ДЛЯ PLAYERA 2 НА ТЕКУЩЕМ STAGEE
                if (CurrentPlayer2Choice == '1')
                {
                    if (r2 == 1)
                    {
                        y2 = CurrentPlayer1Choice;
                    }
                    else //r2 = 0
                    {
                        y2 = CurrentPlayer1Choice == '1' ? '0' : '1';               //Другой
Player на предыдущем Stagee НАОБОРОТ
                    }
                    CurrentPlayer2Choice = (z == 0 && y2 == '0') ? '0' : '1';
                }
                else
                {
                    CurrentPlayer2Choice = (z == 1) ? '1' : '0';
                }


                stage++;


                matrixResult = GetMatrixResult(CurrentPlayer1Choice, CurrentPlayer2Choice,
L, result);

                Console.WriteLine("\r\n\r\nStage {0}: Player1 = {1}, Player2 = {2}. ",
stage, CurrentPlayer1Choice, CurrentPlayer2Choice);
                Console.WriteLine("y1 = {0}, y2 = {1}, z = {2}", y1, y2, z);
                Console.WriteLine(matrixResult);
            }
            #endregion

            Console.WriteLine("\r\n\r\nИтог Игр: {0}; {1}", result[0], result[1]);
        }

        /// <summary>
        /// Парсим ввод пользователем q и S. Закидываем все в лист float
        /// </summary>
        /// <param name="UserInput"></param>
        /// <returns></returns>
        static List<float> ParseInput(string UserInput)
        {
            UserInput = UserInput.TrimStart().TrimEnd();
            List<float> result = new List<float>();
            string[] inputs = UserInput.Split();
            foreach (string input in inputs)
```

```
            {
                float value;
                float.TryParse(input, out value);
                if(value != 0)
                {
                    result.Add(value);
                }
            }

            return result;
        }

        /// <summary>
        /// Парсим ввод пользователем стратегий (усилий) для Playerов
        /// </summary>
        /// <param name="UserInput"></param>
        /// <returns></returns>
        static List<char> ParseInputStrategies(string UserInput)
        {
            UserInput = UserInput.TrimStart().TrimEnd().ToUpper();
            List<char> result = new List<char>();
            string[] inputs = UserInput.Split();
            foreach (string input in inputs)
            {
                char value;
                bool success = char.TryParse(input, out value);
                if(success && (value == '1' || value == '0'))
                {
                    result.Add(value);
                }
            }

            return result;
        }

        /// <summary>
        /// Высчитывание результата Stagea
        /// </summary>
        /// <param name="CurrentPlayer1Choice"></param>
        /// <param name="CurrentPlayer2Choice"></param>
        /// <returns></returns>
        static string GetMatrixResult(char CurrentPlayer1Choice, char CurrentPlayer2Choice,
float L, float[] result)
        {
            string matrixResult = "";
            if (CurrentPlayer1Choice == '1')
            {
                if (CurrentPlayer2Choice == '1')
                {
                    matrixResult = "1,1";
                    result[0] += 1;
                    result[1] += 1;
                }
                else
                {
                    matrixResult = (-L).ToString() + "; " + (1 + L).ToString();
                    result[0] += -L;
                    result[1] += 1+L;
                }
            }
```

```
            else
            {
                if (CurrentPlayer2Choice == '1')
                {
                    matrixResult = (1 + L).ToString() + "; " + (-L).ToString();
                    result[0] += 1+L;
                    result[1] += -L;
                }
                else
                {
                    matrixResult = "0,0";
                    result[0] += 0;
                    result[1] += 0;
                }
            }

            return matrixResult;
        }
    }
}
```

Appendix 2. Program implementation of the second model

```
using System;
using System.Collections.Generic;

namespace Alexandra
{
    class Program
    {
        static void Main(string[] args)
        {

            #region Считывание S
            Console.WriteLine("Введите Количество шагов через пробел");
            string UserInput = Console.ReadLine();
            uint stages = ParseInput(UserInput);

            while (stages == 0)
            {
                Console.WriteLine("Неверный формат! Введите Количество шагов через пробел");
                UserInput = Console.ReadLine();
                stages = ParseInput(UserInput);
            }
            if (stages != 0)
            {
                Console.WriteLine("Количество шагов: S = {0} ", stages);
            }
            #endregion

#region Считывание усилий Playerов
            Console.WriteLine("\r\nEnter the first space-separated signals (y1 =(0,1); y2
=(0,1)) received by player 1 and player 2");
            string UserInputStrategies = Console.ReadLine();
            List<char> inputsStrategies = ParseInputStrategies(UserInputStrategies);
            while(inputsStrategies.Count < 2)
            {
                Console.WriteLine("Wrong format! Enter the first space-separated signals (y1
=(0,1); y2 =(0,1)) received by player 1 and player 2");
                UserInputStrategies = Console.ReadLine();
                inputsStrategies = ParseInputStrategies(UserInputStrategies);
            }
            Console.WriteLine("Entered signals: 1 = {0}, 2 = {1} ", inputsStrategies[0],
inputsStrategies[1]);
            #endregion


            #region Считывание матрицы
            Console.WriteLine("\r\nВведите через пробел A, B, C для матрицы: \r\n AA B0 \r\n
0B CC");
            string UserInputMatrix = Console.ReadLine();
            List<float> inputsMatrix = ParseInputMatrix(UserInputMatrix);
            while (inputsMatrix.Count < 3)
            {
                Console.WriteLine("Неверный формат! Введите через пробел A, B, C для
матрицы: \r\n AA B0 \r\n 0B CC");
                UserInputMatrix = Console.ReadLine();
                inputsMatrix = ParseInputMatrix(UserInputMatrix);
            }
```

```
            Console.WriteLine("Введенная матрица: \r\n {0},{0}   {1},0 \r\n 0,{1}
{2},{2}", inputsMatrix[0], inputsMatrix[1], inputsMatrix[2]);
            #endregion

            #region Инстанс класса рандом
            Random rd = new Random();
            #endregion


            #region Инициализация 1 шага

            int stage = 0;

            // r - вероятность скооперироваться. Высчитывается на каждом шаге
            float A = inputsMatrix[0];
            float B = inputsMatrix[1];
            float C = inputsMatrix[2];
            float r = (C-B) / (A -2*B + C);

            //Первый шаг
            if (rd.Next(0, 101) > r * 100)
            {
                Console.WriteLine("Coop = 1");
            }
            else
            {
                Console.WriteLine("Coop = 0");
            }


            #endregion

            #region 2 и следующие шаги
            //ВТОРОЙ И СЛЕДУЮЩИЙ ШАГИ
            while (stage < stages)
            {
                if (rd.Next(0, 101) > r * 100)
                {
                    Console.WriteLine("Coop = 1");
                }
                else
                {
                    Console.WriteLine("Coop = 0");
                }
                stage++;
            }
            #endregion
        }

        /// <summary>
        /// Парсим ввод пользователем S
        /// </summary>
        /// <param name="UserInput"></param>
        /// <returns></returns>
        static uint ParseInput(string UserInput)
        {
            UserInput = UserInput.TrimStart().TrimEnd();
            List<uint> result = new List<uint>();
            string[] inputs = UserInput.Split();
            foreach (string input in inputs)
```

```
            {
                uint value;
                uint.TryParse(input, out value);
                if(value != 0)
                {
                    result.Add(value);
                }
            }

            return result.Count != 0 ? result[0] : 0;
        }


        /// <summary>
        /// Парсим ввод пользователем матрицы
        /// </summary>
        /// <param name="UserInput"></param>
        /// <returns></returns>
        static List<float> ParseInputMatrix(string UserInput)
        {
            UserInput = UserInput.TrimStart().TrimEnd().ToUpper();
            List<float> result = new List<float>();
            string[] inputs = UserInput.Split();
            foreach (string input in inputs)
            {
                float value;
                bool success = float.TryParse(input, out value);
                if (success)
                {
                    result.Add(value);
                }
            }

            return result;
        }
    }
}
```