Project 4: Q-Learning Dynamics

CID:02535635

January 18, 2024

In this work we will analyse the dynamics of a bimatrix game (A,B), the Q-Learning Dynamic (QLD), which is given by:

$$\frac{\dot{x}_i}{x_i} = (Ay)_i - x^T Ay + T_X \sum_{j=1}^{n} x_j \ln \frac{x_j}{x_i}$$
$$\frac{\dot{y}_i}{y_i} = (x^T B)_i - x^T By + T_Y \sum_{j=1}^{n} y_j \ln \frac{y_j}{y_i}$$

for constants $T_X, T_Y \in \mathbb{R}^{\geq 0}$. For dimensionality reasons, in this setting both players A, B have n pure strategies available, and can mix the pure strategies according to probability vectors: $x(t) = (x_1(t), \ldots, x_n(t))$ and $y(t) = (y_1(t), \ldots, y_n(t))$ respectively. The development of these two vectors is governed by the Q-Learning algorithm and is of primary importance to the rewards gained by both players. A deeper description of its dynamics and the meaning of parameters T_X, T_Y will follow in the answers to the posed questions.

1 Problem

Explain the intuition behind this model. How does it balance the tendency to maximise one's reward against a tendency to 'explore' their actions? The model is fully derived in [1]. It follows the principle of Q-Learning: every player keeps track of their Q-functions: $Q_A(t), Q_B(T)$, and updates them after every turn, according to the principle: at time t, when it is player A's turn, he computes the vector $x(t)$ based on:

$$x_i(t) = \frac{e^{Q_i^X(t)/T_X}}{\Sigma_{j=1}^{n} e^{Q_j^X(t)/T_X}}, \quad i = 1, 2, ..., n$$

computes his Q values at the next step $Q^X(t+1)$ according to

$$\begin{aligned} Q_i^X(t+1) &= Q_i^X(t) + \alpha(r_i^X - Q_i^X(t)) \\ &= (1-\alpha)Q_i^X(t) + \alpha \sum_{j=1}^{n} a_{ij} y_j. \end{aligned}$$

Following the derivation described in [1], it translates to:

$$\dot{Q}_i^X(t) = \alpha(\sum_{j=1}^{n} a_{ij}y_j - Q_i^X(t))$$

so after the same substitution, and scaling the time parameter $t \to \frac{a}{T}t$, we obtain

$$\frac{\dot{x}_i}{x_i} = (Ay)_i - x^T Ay + T_X \sum_{j=1}^{n} x_j \ln \frac{x_j}{x_i}.$$

Similarly we obtain the equation for $\dot{y}_i$.

Agents can decide whether to pursue a more conservative approach (follow the strategies which were profitable in the past), by setting their T parameter close to 0, or to aggressively search for the most profitable strategies, prioritising potential future payoffs, by setting $T >> 0$.

So the main intuitions behind this model are:

- The payoffs for both players follow the bimatrix game scheme.
- The agents follow Q-learning algorithm to assign values (based on past payoffs) to all pure strategies.
- Based on their Q values they adapt their strategies using a parameterised softmax function, which allows them to balance exploration of new strategies and exploitation of previously profitable ones.

2 Problem

Show that interior fixed points of the Q-Learning Dynamic coincide with the Logit Quantal Response Equilibria (QRE). Show that such points always exist.

We will follow the derivation from [2]. The term Quantal Response Equilibrium is used to describe any interior fixed point of the Q-learning dynamics. For positive T_X, T_Y terms, any fixed point must lie in the interior. The intuition behind this, is that for points on the boundary, the exploration term drives the trajectory inward.

Since x_i are greater than 0, we deduce that the RHS of the Q-Learning Dynamics must also be equal to 0. By transforming the system:

$$0 = (Ay)_i + (-x^T Ay + T_X \sum_{j=1}^{n} x_j \ln x_j) - T_X \ln(x_i)$$

$$0 = (x^T B)_i + (-x^T By + T_Y \sum_{j=1}^{n} y_j \ln y_j) - T_Y \ln(y_i) \quad .$$

We notice that the terms in parentheses exist for all i, so let us denote them by K, L for x and y terms respectively. We now take exponential of both sides, and move the terms

with x_i, y_i to the LHS:

$$x_i^{T_X} = e^{(Ay)_i} \cdot e^K$$
$$y_i^{T_X} = e^{(x^T B)_i} \cdot e^L$$

or equivalently:

$$x_i = e^{(Ay)_i/T} \cdot e^{K/T}$$
$$y_i = e^{(x^T B)_i/T} \cdot e^{L/T} \quad .$$

Since the entries x_i are proportional to $\bar{x_i} = e^{(Ay)_i/T}$, we deduce that $e^{K/T}$ is just a normalising term that guarantees $\sum_i x_i = 1$. Therefore:

$$x_i = \frac{\bar{x_i}}{\sum_j \bar{x_j}} = \frac{e^{(Ay)_i/T}}{\sum_j e^{(Ay)_j/T}},$$

and analogously for y_i :

$$y_i = \frac{\bar{y_i}}{\sum_j \bar{y_j}} = \frac{e^{(x^T B)_i/T}}{\sum_j e^{(x^T B)_i/T}}.$$

I am not sure how to get the temperatures out of the denominator in the exponential, but I double checked this derivation and I think it is correct. In [2], section 3.1, where they introduce the term QRE, they also divide the payoffs of pure strategies by the temperature.

These points always exist, for any values of $(Ay)_i, (x^T B)_i$, even for negatives or zero, the exponential function will map it to a positive value. Since there are at least two elements in the sum, the denominator is strictly greater than the numerator, so the components of $\hat{x}, \hat{y}$ are also less than 1. The normalising term in the denominator guarantees that such vectors (x, y) lie within the desired simplices, as the sum of the components must be equal to 1.

3 Problem

Follow the arguments in Kianercy and Galstyan to show that Q-Learning Dynamics is a dissipative system. What does this mean about its behaviour in two-player two-action games?

Following [1], we assume $T_X, T_Y > 0$ and transform the variables:

$$u_k = \ln(\frac{x_{k+1}}{x_1}), \quad v_k = \ln(\frac{y_{k+1}}{y_1})$$

for $k = 1, 2, \ldots, n-1$. We note that we are not loosing any information in the process, since the first system, even though at the beginning the system had 2n equations, because with $\sum_{i=1}^n \dot{x_i} = 0 = \sum_{i=1}^n \dot{y_i}$ it had all its information contained in the first $n-1$ equations. With

$$\tilde{a}_{kj} = a_{k+1,j+1} - a_{1,j+1}, \quad \tilde{b}_{kj} = b_{k+1,j+1} - a_{1,j+1}$$

we obtain:

$$\dot{u_k} = \frac{\sum_{j=1}^{n} \tilde{a}_{kj} e^{v_j}}{1 + \sum_{j=1}^{n} e^{v_j}} - T_X u_k$$
$$\dot{v_k} = \frac{\sum_{j=1}^{n} \tilde{b}_{kj} e^{u_j}}{1 + \sum_{j=1}^{n} e^{u_j}} - T_Y v_k$$

.

Now, we use the Liouville's theorem to argue that the rate of change in volume is proportional to the divergence of RHS. Therefore:

$$\gamma = \sum_{k=1}^{n-1} (\frac{\partial \dot{u_k}}{\partial u_k} + \frac{\partial \dot{v_k}}{\partial v_k}) = -(T_X + T_Y)(n-1) < 0$$

This proves that the system is dissipative. As a result, interior stationary points of the Q-Learning dynamics will be attracting, for positive T.

4 Problem

Show that, if $(\hat{x}, \hat{y}) = (1/n, 1/n)$ (i.e. the uniform distribution) is a Nash Equilibrium, then it is also a QRE for any choice of T_X, T_Y

We first recall what are the proprieties of a Nash Equilibrium in a bimatrix game. Following the definition from the lecture notes, $(\bar{x}, \bar{y})$ is a NE if:

$$\bar{x}^T A \bar{y} \geq x^T A \bar{y}$$
$$\bar{x}^T B \bar{y} \geq \bar{x}^T A y$$

for any $x \in \Delta_X, y \in \Delta_Y$. For $(\hat{x}, \hat{y}) = (1/n, 1/n)$ this means:

$$\frac{1}{n^2} \sum_{i,j} a_{i,j} \geq \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{j=1}^{n} a_{i,j}$$
$$\frac{1}{n^2} \sum_{i,j} b_{i,j} \geq \frac{1}{n} \sum_{j=1}^{n} y_i \sum_{i=1}^{n} b_{i,j} \quad .$$

This is only possible if $\forall k \in [n]: \quad \sum_{j=1}^{n} a_{1,j} = \sum_{j=1}^{n} a_{k,j}$, and $\sum_{i=1}^{n} b_{i,1} = \sum_{i=1}^{n} b_{i,k}$. Meaning, the sums of entries in the rows of A must be equal, as well as the sums of entries in the columns of B. If these sums were not equal, taking a vector $(0, \ldots, 1, \ldots, 0)$, where 1 is in the row/column of the greatest sum, would be greater than $\frac{1}{n^2} \sum_{i,j} a_{i,j}$, which is a contradiction.

When this is the case, we can use the result from the second problem: since $(A\bar{y})_i = (A\hat{y})_j$ for all i, j, the uniform distribution satisfies the first condition of a QRE, and similarly $(\hat{x}^T B)_i = (\hat{x}^T B)_j$ so it also satisfies the second condition, for all T_X, T_Y. Therefore it is a QRE for any choice of temperatures.

5 Problem

Consider the bimatrix game $G = (A, A^T)$ where $G = \begin{bmatrix} 6 & 0 \\ 4 & 2 \end{bmatrix}$. Write down the corresponding Q-Learning Dynamic. Also, find the Nash Equilibria of G and draw the phase plots of QLD in the case $T_X = T_Y = 0$.

We start by noticing that this is a two player game in which both players have only two strategies available. That implies that for the complete analysis of it, we only need to keep track of two values: x_1, y_1, as $x_2 = 1 - x_1$, and $y_2 = 1 - y_2$. We also see that $\dot{x_2} = -\dot{x_1}$, and $\dot{y_2} = -\dot{y_1}$.

To simplify the calculations, let us denote by x, y the values of x_1, y_1 respectively. The vector of strategies will be denoted by $\mathbf{x} = \begin{bmatrix} x \\ 1-x \end{bmatrix}$ and $\mathbf{y}$ respectively.

The QLD system takes the form:

$$\begin{aligned} \frac{\dot{x}}{x} &= 6y - (4xy - 2x + 2y + 2) + T_X(1-x)\ln(\frac{1-x}{x}) \\ &= (1-x) * (4y - 2 + T_X \ln(\frac{1-x}{x})) \end{aligned}$$

$$\begin{aligned} \frac{\dot{y}}{y} &= 6x - (4xy + 2x - 2y + 2) + T_Y(1-y)\ln(\frac{1-y}{y}) \\ &= (1-y) * (4x - 2 + T_Y \ln(\frac{1-y}{y})). \end{aligned}$$

Let us now assume that $T_X = T_Y = 0$. To find the phase portrait of G we first transform the QLD to the form :

$$\begin{aligned} \dot{x} &= x(1-x)(4y-2) \\ \dot{y} &= y(1-y)(4x-2). \end{aligned}$$

It has stationary points when $x, y \in \{0, 1\}$ or $(x, y) = (0.5, 0.5)$. Let us analyse what happens at other points of $\Delta_X \times \Delta_Y$.

For $x \neq 0, 1$, $\dot{x}$ is positive when $4y - 2 > 0$, so for $y \in (0.5, 1]$, it is equal to 0 when $y = 0.5$, and is negative when $y \in [0, 0.5)$

Similarly for $y \neq 0, 1$,

$$\dot{y} \begin{cases} > 0, & \text{if } x \in (0.5, 1] \\ = 0, & \text{if } x = 0.5 \\ < 0, & \text{if } x \in [0, 0.5) \end{cases}$$

Now let us compute the best responses of both agents:

$$\begin{aligned}
\mathrm{BR}_1(\mathbf{y}) &= \arg\max_{\mathbf{x}\in\Delta_X} \begin{bmatrix} x & 1-x \end{bmatrix} \begin{bmatrix} 6 & 0 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} y \\ 1-y \end{bmatrix} \\
&= \arg\max_{\mathbf{x}\in\Delta_X} \begin{bmatrix} x & 1-x \end{bmatrix} \begin{bmatrix} 6y \\ 2+2y \end{bmatrix} \\
&= \begin{cases} \{\begin{bmatrix} 0 \\ 1 \end{bmatrix}\}, \text{if } y < 0.5, \\ \Delta_X, \text{if } y = 0.5, \\ \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\}, \text{if } y > 0.5, \end{cases}
\end{aligned}$$

and analogously:

$$\begin{aligned}
\mathrm{BR}_2(\mathbf{x}) &= \arg\max_{\mathbf{y}\in\Delta_Y} \begin{bmatrix} x & 1-x \end{bmatrix} \begin{bmatrix} 6 & 4 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} y \\ 1-y \end{bmatrix} \\
&= \arg\max_{\mathbf{y}\in\Delta_Y} \begin{bmatrix} 6x & 2+2x \end{bmatrix} \begin{bmatrix} y \\ 1-y \end{bmatrix} \\
&= \begin{cases} \{\begin{bmatrix} 0 \\ 1 \end{bmatrix}\}, \text{if } x < 0.5, \\ \Delta_Y, \text{if } x = 0.5, \\ \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\}, \text{if } x > 0.5. \end{cases}
\end{aligned}$$

We recall from the lectures, that $(\mathbf{x}, \mathbf{y})$ can only be a Nash equilibrium, when $\mathbf{x} \in \mathrm{BR}_1(\mathbf{y})$, and $\mathbf{y} \in \mathrm{BR}_2(\mathbf{x})$. The only points which satisfy this are:

$$(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}), (\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}), (\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}).$$

Due to the analysis of signs of $\dot{x}, \dot{y}$ done before, we expect the first and the third Nash equilibrium to be attracting, while the second one to be unstable.

The hand-drawn portrait takes the form1:

To verify its correctness I also adapted the code from the next exercise, setting $T = 0$ and sampling a large number of initial conditions, we obtain a number of trajectories, which resemble the real phase portrait 2.

6 Problem

Assume that both agents share the same parameter $T_X = T_Y = T$. Produce plots of the trajectories generated by QLD for different choices of T. Describe the behaviour that you observe, producing a bifurcation plot if you wish. Below are plots of a few trajectories for distinct temperatures $T = 0, 0.5, 1, 50, 150$

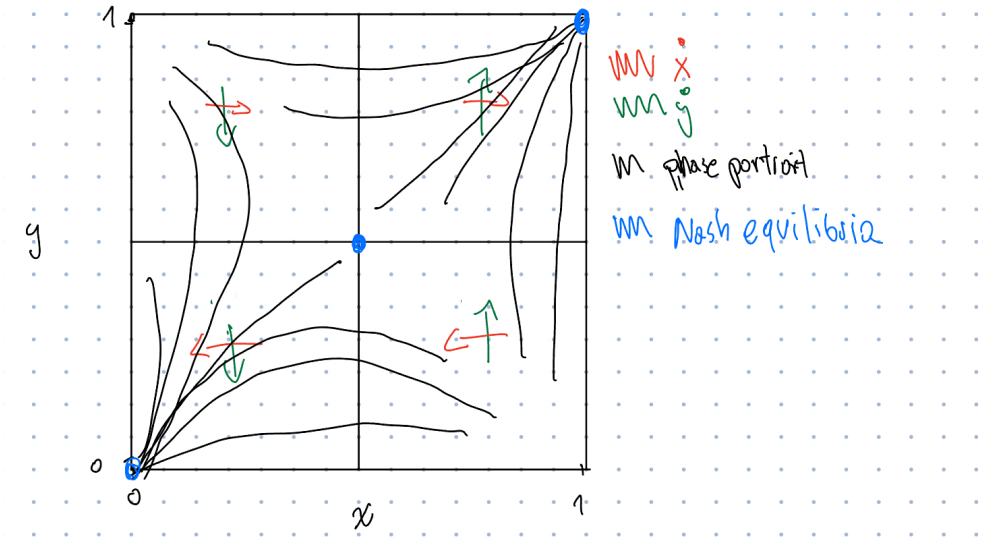

Figure 1: Hand-drawn Portrait

3 is a similar plot to that of 2, just with handpicked initial strategies. We can see the trajectories which begin close to the diagonal of the plot almost reach the QRE, but in the end reach the equilibria corresponding to pure strategies.

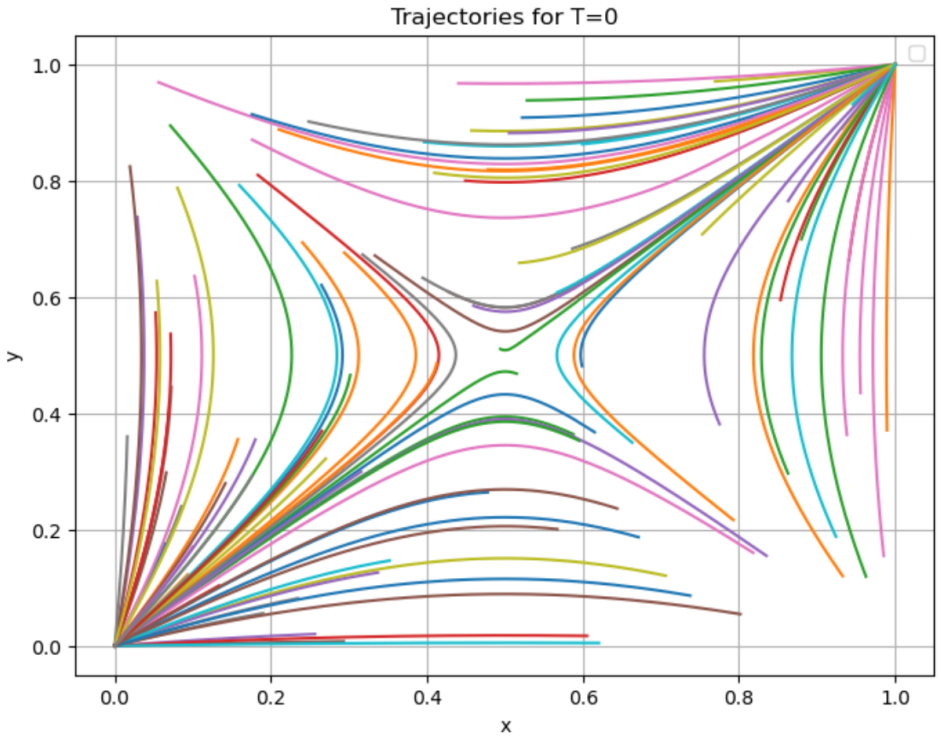

Figure 2: Simulated 100 Trajectories

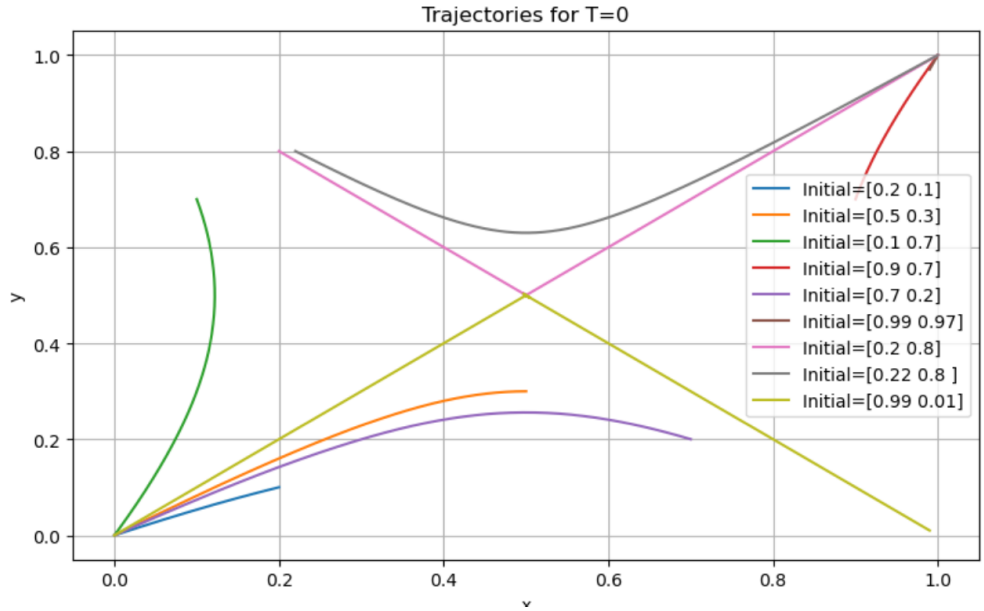

Figure 3: Trajectories for T=0

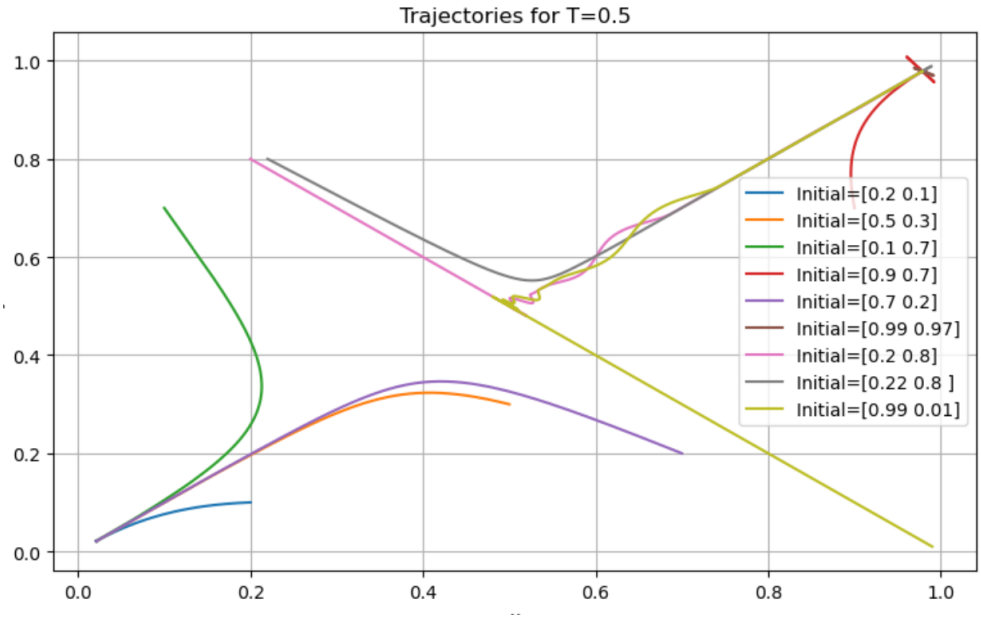

Figure 4: Trajectories for T=0.5

4 changes the dynamics of the points which started near the diagonal. Now they reached the QRE, but ultimately reached the $(\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix})$ strategy. This increased their payoffs, as the entry $a_{1,1} = b_{1,1} = 6$ guarantees the best payoff for both players out of all strategies.

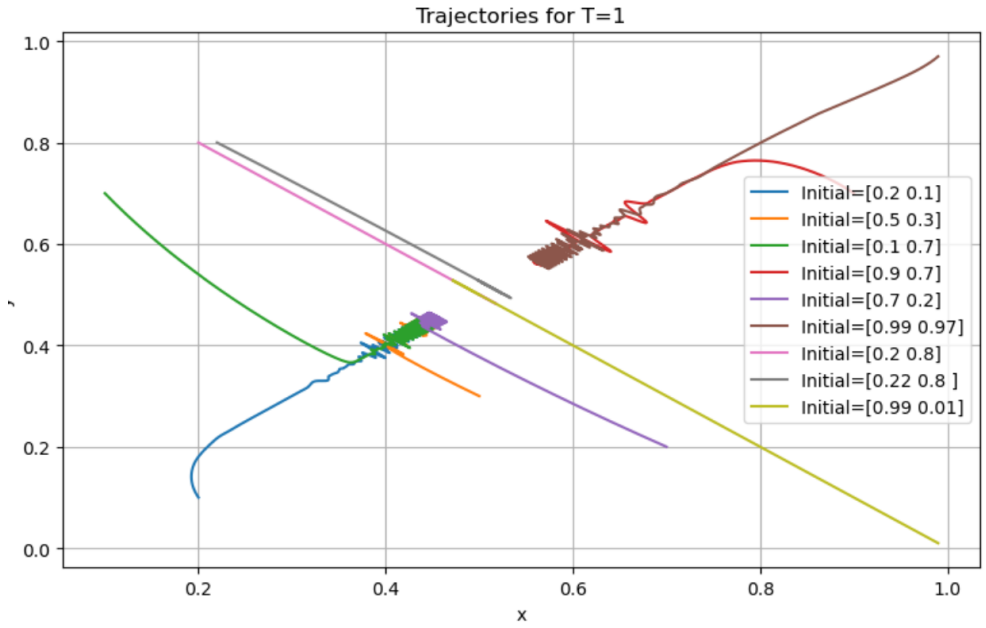

Figure 5: Trajectories for T=1

In this plot 5 we can see the T temperature became sufficiently big for the situation described further in problem 7 to occur: there exists a unique fixed point of the QLD. We can see that even the path which began at $(\begin{bmatrix}0.99\\0.01\end{bmatrix}, \begin{bmatrix}0.97\\0.03\end{bmatrix})$ in the process of exploration, landed at the uniform distribution, corresponding to the QRE, instead of the 'closer' Nash Equilibrium corresponding to the pure strategies $(\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix})$.

Most paths follow the diagonal connecting $(0,0)$ and $(1,1)$, and we can see slight oscillations around it - they correspond to the exploration around the last strategies.

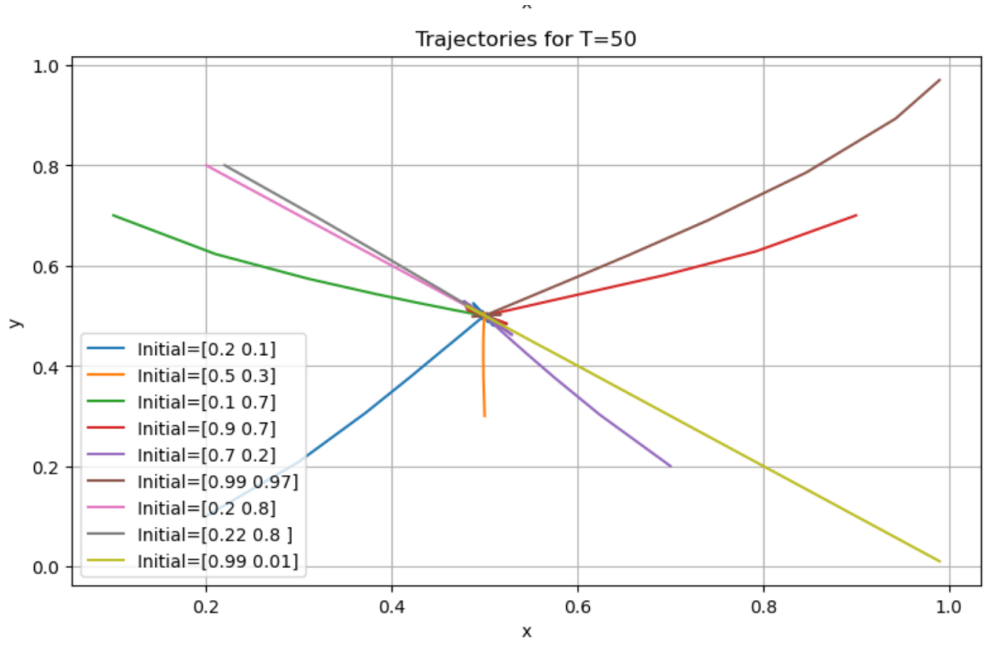

Figure 6: Trajectories for T=50

On 6 we can see that all trajectories immediately start approaching the QRE.

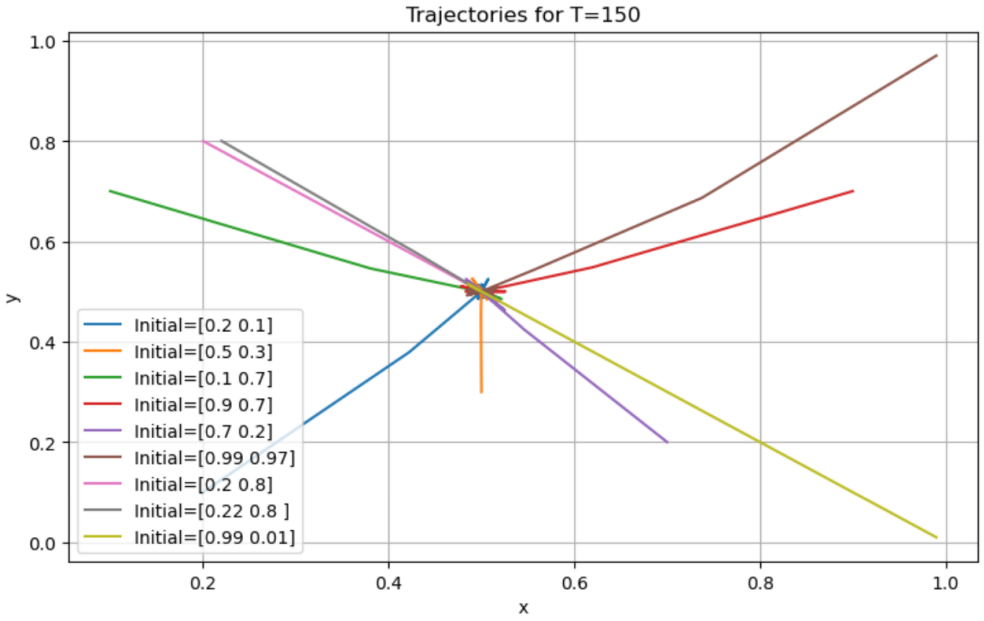

Figure 7: Trajectories for T=150

Again, as for $T = 50$, all trajectories reach the QRE, we can see the oscillations around it, which can be understood as exploration attempts by the players.

As described in [1] the trajectories converge to three rest points for small values of T, and for temperatures greater than the critical value T_C to one rest point. The game G is an example of a coordination game, therefore we can follow the analysis of [1]and compute:

$$a = c = \frac{4}{T}, \quad b = d = \frac{-2}{T}$$

so the condition $-1 = \frac{b}{a} + \frac{d}{c}$ is satisfied, and the game is not strictly risk dominant, since for the uniform distribution $\mathbf{y} = (0.5, 0.5)$, the payoff of the first player in case of a pure strategy $\bar{\mathbf{x}} = (1, 0)$ is equal to the payoff of the uniformly mixed strategy:

$$\bar{\mathbf{x}}^T A\mathbf{y} = \bar{\mathbf{x}}^T \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 1 * 3 + 0 * 3 = 3 = 0.5 * 3 + 0.5 * 3 = \hat{\mathbf{x}} A\mathbf{y}.$$

Therefore the rest point structure will follow a continuous pitchfork transformation. We can also analyse the $g(u)$ function defined in this paper.

I attempted to plot the bifurcation plot, but my understanding of this is not sufficient yet (I am taking the module in the Spring semester). Analysis of the trajectories generated for $T = 0.99, 1, 1.01, 1.02$ suggests the critical temperature T_C is equal to 1, and the plot should look like Figure 8.

7 Problem

Continuing with the game G, for what value of T is there a unique fixed point of QLD? Argue that this unique fixed point must be globally attracting.

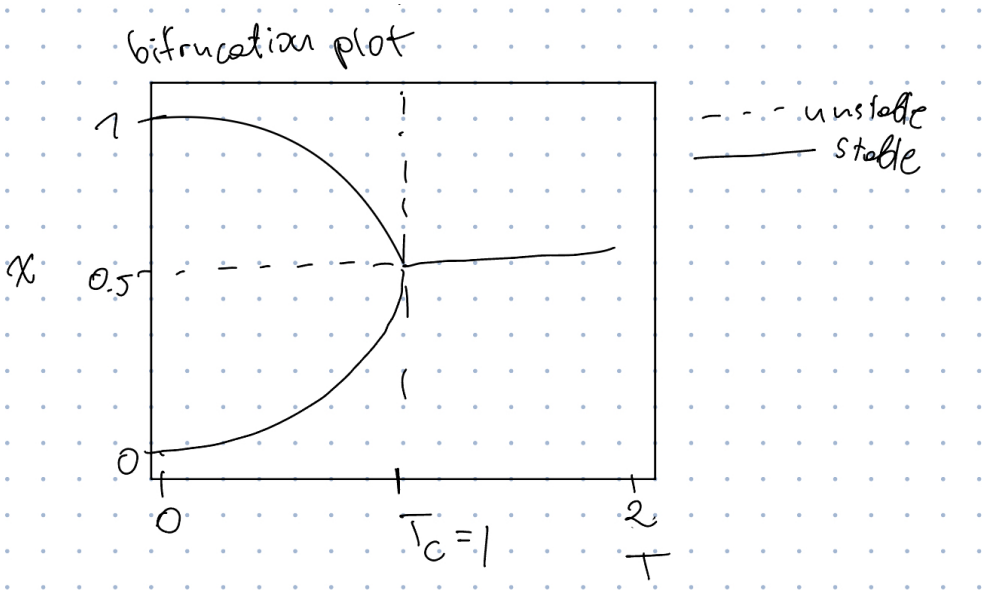

Figure 8: Approximate Bifurcation Plot

The first part of the question is already partially answered in the previous section. We experimentally found this value to be $T_C = 1$. We can also analyse this with the $g(u)$ function as in [1].

The unique fixed point must be globally attracting, due to the dissipative property of the system. As described in the third problem.

8 Problem

Consider the game presented in Sato et al. This is a bimatrix game (A, B) in which:

$$A = \begin{pmatrix} \epsilon_X & -1 & 1 \\ 1 & \epsilon_X & -1 \\ -1 & 1 & \epsilon_X \end{pmatrix} \quad B = \begin{pmatrix} \epsilon_Y & 1 & -1 \\ -1 & \epsilon_Y & 1 \\ 1 & -1 & \epsilon_Y \end{pmatrix}$$

Find its Nash Equilibrium $(\hat{x}, \hat{y})$.

We recall the definition of a Nash equilibrium: $(\hat{x}, \hat{y}) \in \Delta_X \times \Delta_Y$ is a Nash Equilibrium of the game (A, B) iff:

$$\hat{x}^T A \hat{y} \geq x^T A \hat{y} \quad \forall x \in \Delta_X$$
$$\hat{x}^T B \hat{y} \geq \hat{x}^T B y \quad \forall y \in \Delta_Y$$

or equivalently $(\hat{x}, \hat{y}) \in \mathrm{BR}_1(\hat{y}) \times \mathrm{BR}_2(\hat{x})$

We will first show that the uniform distribution is a NE of this system. Let us consider $(\hat{x}, \hat{y}) = ((\frac{1}{3})_3, (\frac{1}{3})_3)$ We calculate the best response for the first player assuming the second

player plays $\hat{y}$:

$$\begin{aligned}\mathrm{BR}_1(\hat{y}) &= \arg\max_{x\in\Delta_X} x^T A\hat{y} \\ &= \arg\max_{x\in\Delta_X} x^T \begin{bmatrix}\epsilon_X \\ \epsilon_X \\ \epsilon_X\end{bmatrix} \\ &= \Delta_X\end{aligned}$$

and analogously we can see that

$$\mathrm{BR}_2(\hat{x}) = \Delta_Y$$

, so we can clearly see that the condition

$$(\hat{x},\hat{y}) \in \mathrm{BR}_1(\hat{y}) \times \mathrm{BR}_2(\hat{x})$$

holds and therefore $(\hat{x},\hat{y})$ is indeed a Nash Equilibrium.

To deduce that there are no more Nash Equilibria, let us consider a pair $(\tilde{x},\tilde{y})$ as a candidate, assuming it is not the uniform distribution. This implies that one of the coordinates of $\tilde{y}$ is greater equal than the other two.

Let us assume that it is strictly greater. By the symmetry in this game we can assume that it is the first coordinate. Then the best response of the first player has to put more weight on the pure strategy that directly wins with the first strategy of the second player, so it will put more weight on the second coordinate.

This implies that the point $\tilde{y}$ can not be the best response for the second player, as he should have put more weight on the strategy that beats the second strategy of the first player: the third pure strategy. This contradiction shows that there cannot be a Nash Equilibrium where one strategy has more weight than the other two.

To prove that there are no strategies where one of the pure strategies has less weight than the remaining two, we can work in a setting where $\tilde{y}_1 = \tilde{y}_2 > \tilde{y}_3$. The best response of the first player is then $< e_2, e_3 >$, but the best response of the second player to any strategy in this set has to lie in $< e_1, e_3 >$, which is a contradiction to the assumption that the first and the second strategies were dominating.

We conclude that the uniform distribution is the only Nash Equilibrium of this game.

9 Problem

We will first consider the case $\epsilon_X = \epsilon_Y = 0$. Assume also that $T_X = T_Y = 0$. Show that

$$D_{KL}(\hat{x}\|x) + D_{KL}(\hat{y}\|y)$$

is a constant of the motion, where D_KL denotes the KL-Divergence between discrete probability distributions (look up its definition). What does this constant of motion tell us about the stability of $(\hat{x},\hat{y})$?

The KL-Divergence on a discrete space of size three (as is the case for the probability vectors in Δ_X , and Δ_Y) is defined as:

$$D_{KL}(p||q) = \sum_{i=1}^{3} p_i \dot{\ln} \frac{p_i}{q_i}$$

The proof will follow the method shown in [3], and is based on the property that $A = -B$. We define $H = D_{KL}(\hat{x}\|x) + D_{KL}(\hat{y}\|y)$ and simply compute its time derivative. In the process we will use the following:

$$\sum_{i=1}^{3} \hat{x_i} \frac{\dot{x_i}}{x_i} = \sum_{i=1}^{3} \hat{x_i}((Ay)_i - x^T Ay) = \sum_{i=1}^{3} \hat{x_i}(Ay)_i - (\sum_{i=1}^{3} \hat{x_i}) x^T Ay = \hat{x}^T Ay - x^T Ay$$

,where we used the QLD dynamics to substitute for $\frac{\dot{x_i}}{x_i}$, and analogously

$$\sum_{i=1}^{3} \hat{y_i} \frac{\dot{y_i}}{y_i} = x^T B\hat{y} - x^T By \quad .$$

Therefore we obtain:

$$\frac{dH}{dt} = -\sum_{i=1}^{3} \hat{x_i} \frac{\dot{x_i}}{x_i} - \sum_{i=1}^{3} \hat{y_i} \frac{\dot{y_i}}{y_i} = -(\hat{x}^T Ay - x^T Ay) - (x^T B\hat{y} - x^T By)$$

We now notice, A multiplied by the uniform distribution gives the zero vector, and therefore

$$(\hat{x} - x)^T A\hat{y} = (\hat{x} - x)^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

and similarly:

$$\hat{x}^T B(\hat{y} - y) = 0$$

Therefore we can add these to $\frac{dH}{dt}$ without changing its value. These two terms will allow us to factorise the term in a clear way:

$$\begin{aligned} \frac{dH}{dt} &= -(\hat{x}^T Ay - x^T Ay) - (x^T B\hat{y} - x^T By) \\ &= -(\hat{x} - x)^T Ay - x^T B(\hat{y} - y) \\ &= (\hat{x} - x)^T A\hat{y} - (\hat{x} - x)^T Ay + \hat{x}^T B(\hat{y} - y) - x^T B(\hat{y} - y) \\ &= (\hat{x} - x)^T A(\hat{y} - y) + (\hat{x} - x)^T B(\hat{y} - y) \\ &= (\hat{x} - x)^T (A + B)(\hat{y} - y) \\ &= 0 \quad , \end{aligned}$$

since in our game $A + B = 0$. Therefore the value H is preserved during the motion.

This constant tells us that for this specific case the motion will not approach the Nash Equilibrium of the system, unless at the beginning $H = 0$, since it would mean that the value of H had to change.

10 Problem

Explore numerically the behaviour of Q-Learning of various choices of $T_X = T_Y = T \in [0, 1)$ and for different choices of $\epsilon_X, \epsilon_Y \in [-0.1, 0.1]$. Discuss the behaviour that you would expect and what you observe. Feel free to take inspiration from Sato et al. to guide your own exploration.

The plots I generated here in the first version of the file were quite misleading. I edited the code in two places: in definition of the $player_B$, and in the indices of $xx[]$ which were plotted on the graphs. Luckily the analysis was mostly correct, however, the last plot, where the smallest value of T for which the trajectories converge to the inner NE was incorrect. The main difference was noticeable in the case of $T = 0.1, \epsilon_X = \epsilon_Y = -0.1$, where with the correct labels, we observe the convergence to the NE, as expected from the dissipative nature of the system.

Similarly to the coordination game, we simulated in the previous problem, I expect a more unstable motion for small values of T. It was also discussed in [3], where the parameter α was equivalent to T, and chaotic behaviour was observed for $T = 0$, $\epsilon_X = -\epsilon_Y = 0.5$.

A repeated version of this simulation for parameters $T = 0$, $\epsilon_X = -\epsilon_Y = 0.1$ yields:

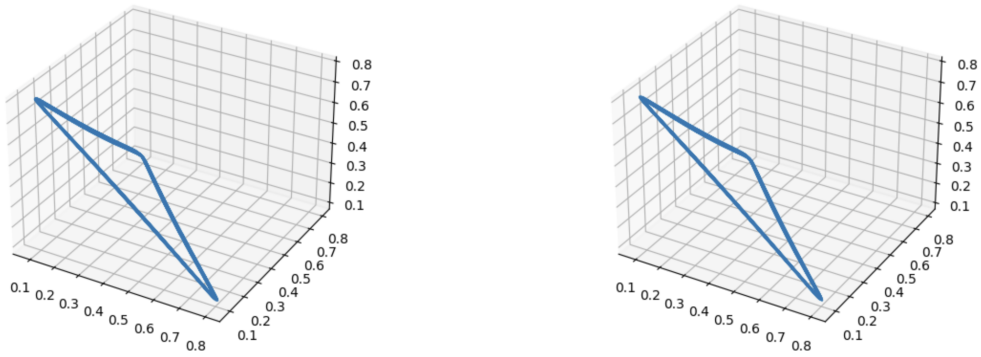

Figure 9: $T = 0$, $\epsilon_X = -\epsilon_Y = 0.1$

We observe in 9 pseudo-periodic behaviour, although the amplitude of fluctuations appears to be less chaotic than in [3].

Let us simulate a game more similar to a coordination game: $T = 0$, $\epsilon_X = \epsilon_Y = 0.1$

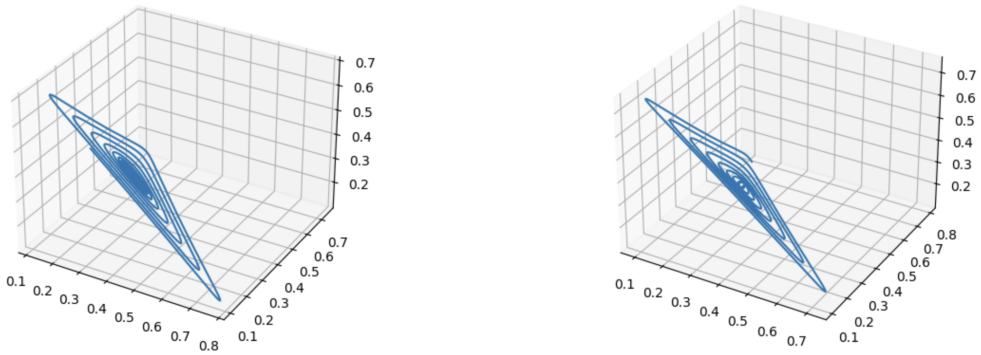

Figure 10: $T = 0$, $\epsilon_X = \epsilon_Y = 0.1$

We can now see 10 that the trajectory is attracted to the Nash Equilibrium, even though the exploration parameter remains set to 0.

What if the game penalises draws? We simulate $T = 0$, $\epsilon_X = \epsilon_Y = -0.1$. The intuition suggests that the players will avoid mixed strategies, and therefore move on the boundaries of their simplices.

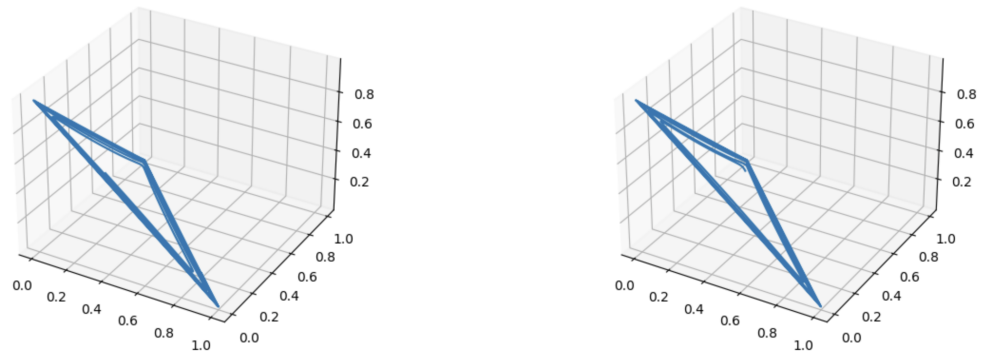

Figure 11: $T = 0$, $\epsilon_X = \epsilon_Y = -0.1$

And indeed, on figure 11 we observe that the trajectories avoid the centre of the simplex. We note, that in this way the players will not reach any fixed point.

Let us now increase the temperature parameter to $T = 0.1$ and observe how exploration influences the dynamics for the four cases of $\epsilon_X = \epsilon_y = 0$, $\epsilon_X = \epsilon_Y = 0.1$, $\epsilon_X = -\epsilon_Y = 0.1$ and $\epsilon_X = \epsilon_Y = -0.1$

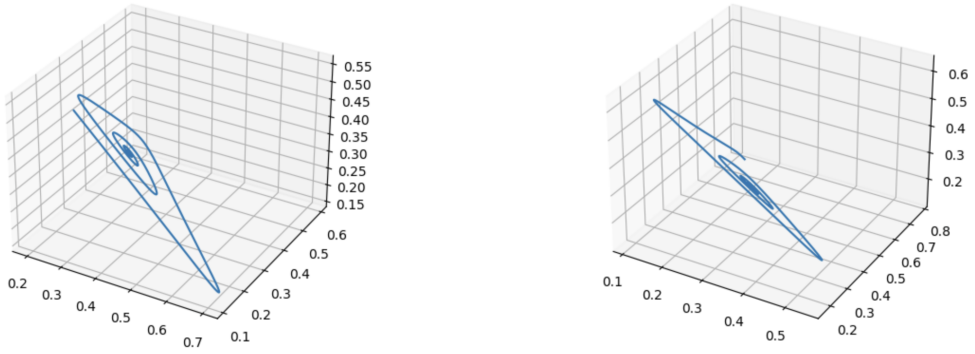

Figure 12: $T = 0.1$, $\epsilon_X = \epsilon_Y = 0$

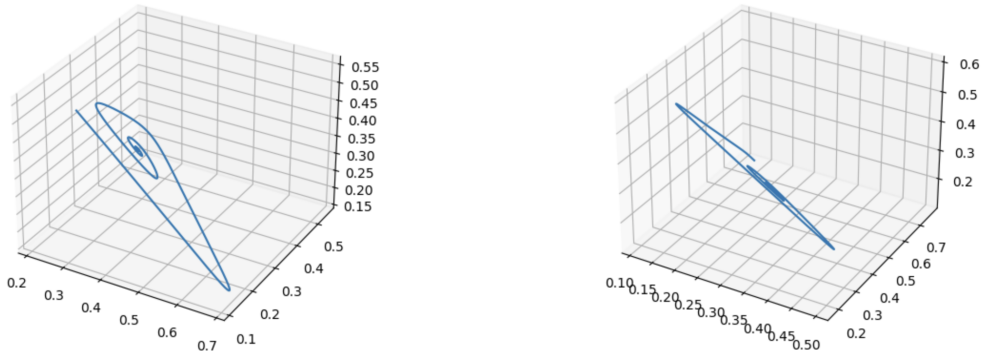

Figure 13: $T = 0.1$, $\epsilon_X = \epsilon_Y = 0.1$

This time all plots 12, 13, 14, 15 show similar behaviour: convergence towards the Nash Equilibrium.

Since the case when draws were penalised, that is $\epsilon_X = \epsilon_Y = -0.1$ was the most interesting, I wanted to check for which temperatures the trajectory will start to approach the Nash Equilibrium. Naturally, previous problem guarantees that the system is dissipative, and therefore every trajectory for positive temperatures should converge to the QRE, but I did not observe this effect for temperatures close to zero.

In the correctly labeled plots, the convergence to the interior NE was observable for $T = 0.04$, as shown on fig. 16.

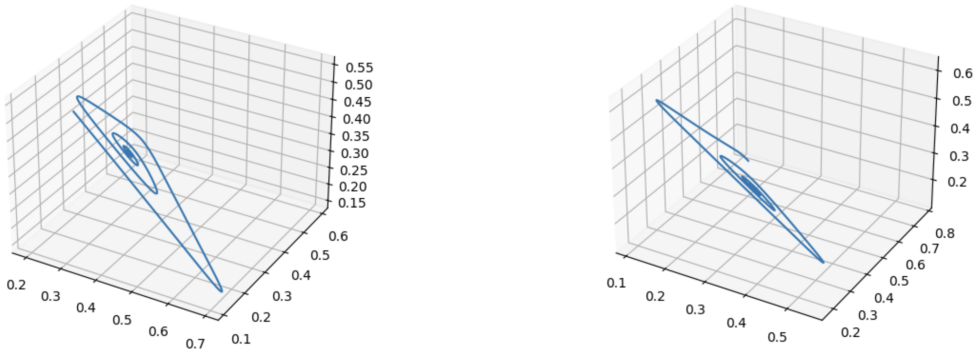

Figure 14: $T = 0.1$, $\epsilon_X = -\epsilon_Y = 0.1$

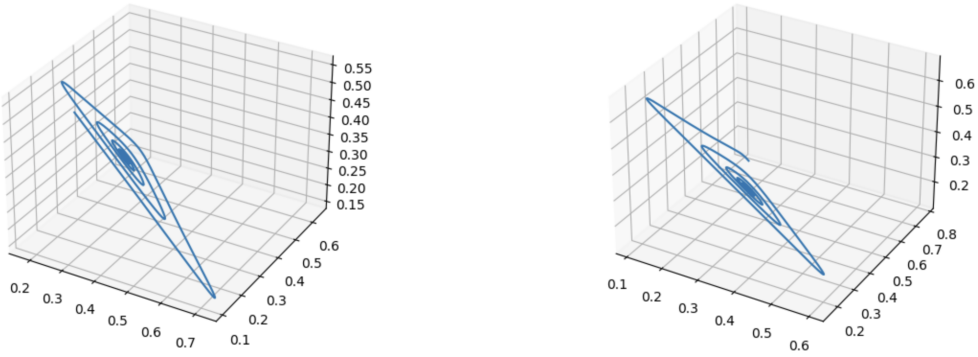

Figure 15: $T = 0.1$, $\epsilon_X = \epsilon_Y = -0.1$

11 Problem

Compute the average payoffs over time along orbits. How do these compare with the payoffs received if the agents were to play the Nash Equilbrium at each step?

We note that the payoffs at Nash Equilibrium are equal to

$$p_1 = 1/3 * \epsilon_X \quad p_2 = 1/3 * \epsilon_Y$$

For trajectories which converge to the Nash Equilibrium the average payoff, by the law of large numbers will converge to p_1, p_2 for both players. And indeed, for positive T, we observe this experimentally.

For example, for $T = 0.1$, $\epsilon_X = \epsilon_Y = 0.1$, the average payoffs are: 0.034130.03413. For $T = 0.1$, $\epsilon_X = -\epsilon_Y = 0.1$ the payoffs are: $(0.0336 - 0.0336)$. Both of them are larger in absolute value than the (p_1, p_2) values, but this might be dependant on the initial condition of the system.

The unique case of penalised draws: $T = 0$, $\epsilon_X = \epsilon_Y = -0.1$, when the trajectory does not converge to the QRE has average payoffs: $(-0.021, 0.012)$, which is better, than the average payoff at the NE.

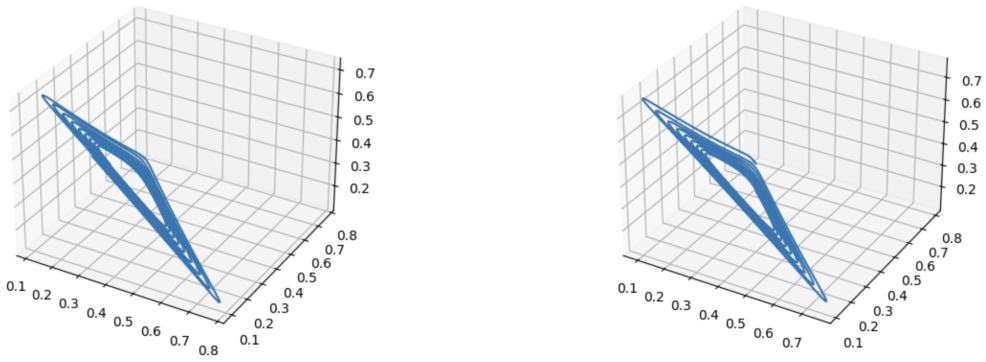

Figure 16: $T = 0.04$, $\epsilon_X = \epsilon_Y = -0.1$

A Appendix A: Code for the problems in the main part

A.1 Phase portrait

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint, solve_ivp, ode
import time as tm

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Define the replicator function
def replicator(t, x, A, B, T):
    x1, y1 = x
    dx = np.zeros(2)
    dx[0] = x1 * (1 - x1) * (4 * y1 - 2) + T * x1 * (1 - x1) * np.
       log( (1-x1) /x1)
    dx[1] = y1 * (4 * x1 - 2) * (1 - y1) + T * y1 * (1 - y1) * np.
       log( (1-y1) / y1)
    return dx

# Set up parameters
A = np.array([[6, 0], [4, 2]])
B = A.T

# Time settings
t_step = 0.01
t_final = 10
time = np.arange(0, t_final, t_step)


# Single value of T
T_value = 0

# Initial values, 100 randomised in two dimensions; one for x, one
   for y
initial_values = np.zeros([100,2])
for i in range(100):
    initial_values[i,:] = np.random.uniform(0,1,2)

# Plot the portrait
plt.figure(figsize=(8, 6))

for z0 in initial_values:
    sol = solve_ivp(replicator, [0, t_final], y0=z0, args=(A, B,
       T_value), method='DOP853', t_eval=time)
    xx = sol.y
    plt.plot(xx[0, :], xx[1, :])
```

```
43
44 # Plot config.
45 plt.title(f'Trajectories for T={T_value}')
46 plt.xlabel('x')
47 plt.ylabel('y')
48 plt.legend()
49 plt.grid(True)
50 plt.show()
```

Listing 1: phase portrait of the cooperation game

A.2 Trajectories

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Define the replicator function
def replicator(t, x, A, B, T):
    x1, y1 = x
    dx = np.zeros(2)
    dx[0] = x1 * (1 - x1) * (4 * y1 - 2) + T * x1 * (1 - x1) * np.
       log( (1-x1) /x1)
    dx[1] = y1 * (4 * x1 - 2) * (1 - y1) + T * y1 * (1 - y1) * np.
       log( (1-y1) / y1)
    return dx

# Set up parameters; not used ultimately because the computed form
   is earsier to read
A = np.array([[6, 0], [4, 2]])
B = A.T

# Time settings
t_step = 0.01
t_final = 100
time = np.arange(0, t_final, t_step)

# Initial values
initial_values = [
    np.array([0.2, 0.1]),
    np.array([0.5, 0.3]),
    np.array([0.1, 0.7]),
    np.array([0.9, 0.7]),
    np.array([0.7, 0.2]),
    np.array([0.99, 0.97]),
    np.array([0.2, 0.8]),
    np.array([0.22, 0.8]),
    np.array([0.99, 0.01]),
]

# T values
T_values = [  0.5 , 1, 5 , 50 ,150]

# Plots; we keep the results for different values of T on different
   plots.
fig, axes = plt.subplots(nrows=len(T_values), ncols=1, figsize=(8, 5
    * len(T_values)))

for i, T_value in enumerate(T_values):
```

```
    for z0 in initial_values:
        sol = solve_ivp(replicator, [0, t_final], y0=z0, args=(A, B,
            T_value), method='DOP853', t_eval=time)
        xx = sol.y
        axes[i].plot(xx[0, :], xx[1, :], label=f'Initial={z0}')

    # Customize each subplot
    axes[i].set_title(f'Trajectories for T={T_value}')
    axes[i].set_xlabel('x')
    axes[i].set_ylabel('y')
    axes[i].legend()
    axes[i].grid(True)

plt.tight_layout()
plt.show()
```

Listing 2: Trajectories depending on the temperature

B Appendix B: Code for the mastery material

```
1
2  # The following code is heavily expired by the code I found in the
       appendix to the lecture notes.
3  # I adapted the replicator to mimic the QLD, but most of it remained
       the same.
4  # It was titled 'replicatorRPS - Sato, so it was probably also used
       to produced the plots found in the referenced paper
5  # by Sato et al.
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  #import networkx as nx
10 from scipy.integrate import odeint, solve_ivp, ode
11 from mpl_toolkits.mplot3d import Axes3D
12 from scipy import sparse
13 import time as tm
14 from scipy.spatial.distance import pdist, squareform
15 from mpl_toolkits.axes_grid1 import make_axes_locatable
16
17 def replicator(t, x, A, B, T):
18
19     boltzmann_term = np.zeros(6)
20     for i in range(3):
21         for j in range(3):
22             if i != j:
23                 boltzmann_term[i] += x[j] * np.log(  x[j] / x[i])
24     for i in range(3,6):
25         for j in range(3,6):
26             if i != j:
27                 boltzmann_term[i] += x[j] * np.log( x[j] / x[i] )
28
29     #print(boltzmann_term)
30     dx = np.zeros(6)
31     dx[:3] = x[:3] * (A @ x[3:] - np.transpose(x[:3]) @ (A @ x[3:])
           + T * boltzmann_term[:3])
32     dx[3:] = x[3:] * (B.T @ x[:3] - np.transpose(x[:3]) @ (B @ x
           [3:]) +  T * boltzmann_term[3:])
33     return dx
34 # Initial value
35 z0 = [0.25, 0.1, 0.65, 0.15, 0.75, 0.1]
36 # or = np. random.uniform(0.1, 0.2, 6).T
37 # Time parameters
38 t_step = 0.03
39 t_final = 100
40 time = np.arange(0, t_final, t_step)
```

```
41
42 #exploration parameter, temperature
43 T = 0.04
44 # Define matrices
45 epsilonx = -0.1
46 epsilony = epsilonx
47
48 A = np.array([
49     [epsilonx, 1, -1],
50     [-1, epsilonx, 1],
51     [1, -1, epsilonx]
52 ])
53
54 BSato= np. array([[epsilony, 1 ,-1],
55                   [-1, epsilony ,1],
56                    [ 1, -1 ,epsilony]])
57 B=BSato.T
58 #print ('A=' ,A)
59 #print('BSato=',BSato)
60 #print ('B=',B)
61 # integrate ODE
62 sol = solve_ivp(replicator, [0,t_final], y0=z0, method='DOP853',
      t_eval=time, args=(A,B,T))
63 xx = sol.y
64
65 playerA = xx[:3,:]
66 playerB = xx[3:,:] # here there was a bug!!!
67 sum1 = 0
68 sum2 = 0
69 for i in range(playerA.shape[1]):
70     sum1 += (np.transpose(playerA)[i,:] @ A @ playerB[:,i])
71     sum2 += (np.transpose(playerA)[i,:] @ B @ playerB[:,i])
72
73
74
75 #print(np.sum(np.transpose(playerA) @ A @ playerB) )
76 av_payoff_1 = sum1/ playerA.shape[1]
77 av_payoff_2 = sum2 / playerA.shape[1]
78
79 print('average payoffs ' , av_payoff_1 , av_payoff_2)
80
81
82 transient = 10
83 #check whether the vectors stored in x remain probability vectors:
84 #s = np.zeros(xx.shape[1])
85 #s = xx[0,:] + xx[1,:] + xx[2,:]
86 #print(xx)
```

```
#print(50*'-')
#plt.plot(s)
#plt.plot (xx[4, :])
#plt.plot (xx[5, :])
plt.figure (figsize=(15,5))
plt.subplot (121, projection='3d')
plt.plot(xx[0,transient:], xx[1,transient:], xx[2,transient:]) #here I was plotting the wrong indices
plt.subplot (122, projection='3d')
plt.plot(xx[3,transient:], xx[4,transient:], xx[5,transient:]) #here I also fixed the indices.
```

Listing 3: Trajectories of the R-P-S game

References

[1] Aram Galstyan Ardeshir Kianercy. Dynamics of boltzmann q learning in two-player two-action games. 2018.

[2] Kelly Spendlove Stefanos Leonardos, Georgios Piliouras. Exploration-exploitation in multi-agent competition: Convergence with bounded rationality.

[3] E. Akiyama J. P. Crutchfield Y. Sato. Stability and diversity in collective adaptation.