AGTA - Coursework 1

Exam number: B122217

February 2021

1 Question 3

Consider the following simple 2-player zero-sum game where the payoff table for Player 1 (the row player) is given by:

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

We can view this as a game where each player chooses "heads" (H) or "tails" (T), where the first strategy for each player is denoted H and the second strategy is denoted T.

1.1 Question 3a

First, what is the unique Nash equilibrium, or equivalently the unique minimax profile of mixed strategies for the two players, in this game? And what is the minimax value of this game?

In Questions 3a and 3b I will be exclusively answering them with regards to game B.

To find the Nash equilibrium of this game we can use the the corollary of the Nash theorem, if player 2 is playing against player 1's mixed strategy, both of player 2's pure strategies must be a best response to player 1. Therefore, player 2 expected value should be the same for either pure strategy it plays against players 1 mixed strategy, and vice versa.

Now suppose player 1 plays row 1 with probability p and row 2 with probability 1-p. Likewise player 2 plays column 1 with probability q and column 2 with probability 1-q.

Then using the corollary to Nash theorem we get:

$$\begin{gathered} -3p + 0(1-p) = -p + -2(1-p) \\ -3p = p - 2 \\ 2 = 4p \\ p = \frac{1}{2}, 1 - p = \frac{1}{2} \end{gathered}$$

Likewise for player 2 we get:

$$
\begin{aligned}
3q + 1(1-q) &= 0q + 2(1-q) \\
3q + 1 - q &= 2 - 2q \\
2q + 1 &= 2 - 2q \\
4q &= 1 \\
q &= \frac{1}{4}, 1 - q = \frac{3}{4}
\end{aligned}
$$

So we get an expression for the minimax profile: $x = ((\frac{1}{2}, \frac{1}{2}), (\frac{1}{4}, \frac{3}{4}))$, in this case player 1 mixed strategy x_1^* is $(\frac{1}{2}, \frac{1}{2})$ and players 2 mixed strategy x_2^* is $(\frac{1}{4}, \frac{3}{4})$.

To find the minimax value of this game, we can use the Minimax Theorem where we have that $v^* = (x_1^*)^T A x_2^*$.Therefore this is equal to:

$$
(x_1^*)^T A x_2^* = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & \frac{3}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \end{bmatrix} = \frac{3}{2}\frac{1}{4} + \frac{3}{2}\frac{3}{4} = \frac{3}{8} + \frac{9}{8} = \frac{3}{2}
$$

So we have that the minimax value of the game equal to $\frac{3}{2}$.

1.2 Question 3b

Prove that, regardless how the two players start playing the game in the first round, the "statistical mixed strategies" of both players in this method of repeatedly playing the matching pennies game will, in the long run, as the number of rounds goes to infinity, converge to their mixed strategies in the unique Nash equilibrium of the game. You are allowed to show that this holds using any specific tie breaking rule that you want. Please specify the precise tie breaking rule you have used. (It turns out that it hold true for any tie breaking rule. But some tie breaking rules may make the proof easier than others.)

We need show that the "statistical mixed strategies" of both players in this method of repeatedly playing the game will, in the long run, as the number of rounds goes to infinity, converge to their mixed strategies in the unique Nash equilibrium of the game. It is important to note, that each player i accumulates statistics on how its opponent has played until now, meaning how many Heads and how many Tails have been played by the opponent, over all rounds of the game played thus far. So in this case we have define that N Heads and M Tails.

Therefore, in this game we have that the statistical mixed strategies are given by:

$$\sigma = \left(\frac{N}{(N+M)}, \frac{M}{(N+M)} \right)$$

I will denote σ_1 as the statistical mixed strategies that Player 2 calculates, since this is what they calculate Player 1 to play. Likewise, σ_2 is the statistical mixed strategies that Player 1 calculates.

We know, form question 3a, that the unique Nash equilibrium for this problem is

$$x = \left(\left(\frac{1}{2}, \frac{1}{2} \right), \left(\frac{1}{4}, \frac{3}{4} \right) \right)$$

Therefore, we need to show that $x_1 = \left(\frac{1}{2}, \frac{1}{2}\right) = \sigma_1$ as the number of rounds goes to infinity. Likewise, we need to show that $x_2 = \left(\frac{1}{4}, \frac{3}{4}\right) = \sigma_2$ as the number of rounds goes to infinity.

To show this, I will do this experimentally. I have created a function called "run" which lets you run a simulation of the game over a given number of rounds you want, for a given 2x2 player game and the starting positions for Player 1 and Player 2 . I have attached the function at the end of this question. We note that Player 1 and Player 2 can begin in any of their pure strategies $(0, 1)$ or $(1, 0)$. Therefore I will show graphically that for all combinations of the starting positions, they do indeed convergence to the NE x.

I will first show this by giving a table with the number of rounds in the simulation and the final value of σ calculated by both players after the given number of rounds. I will do this for all possible starting strategies for Player 1 and 2.

Starting position Player 1: [1,0] and Player 2: [1,0]		
Rounds	σ_1	σ_2
10	[0.4, 0.6]	[0.3, 0.7]
50	[0.54, 0.46]	[0.34, 0.66]
100	[0.47, 0.53]	[0.27, 0.73]
250	[0.536, 0.464]	[0.228, 0.772]
500	[0.542, 0.458]	[0.242, 0.758]
1000	[0.518, 0.482]	[0.262, 0.738]
2500	[0.5196, 0.4804]	[0.2468, 0.7532]
5000	[0.5148, 0.4852]	[0.2482, 0.7518]
10000	[0.4968, 0.5032]	[0.2517, 0.7483]
25000	[0.50144, 0.49856]	[0.24664, 0.75336]
50000	[0.50248, 0.49752]	[0.24834, 0.75166]

Starting position Player 1: [1,0] and Player 2: [0,1]		
Rounds	σ_1	σ_2
10	[0.8, 0.2]	[0.3, 0.7]
50	[0.54, 0.46]	[0.2, 0.8]
100	[0.52, 0.48]	[0.21, 0.79]
250	[0.504, 0.496]	[0.22, 0.78]
500	[0.506, 0.494]	[0.272, 0.728]
1000	[0.528, 0.472]	[0.253, 0.747]
2500	[0.4968, 0.5032]	[0.2444, 0.7556]
5000	[0.5052, 0.4948]	[0.255, 0.745]
10000	[0.5076, 0.4924]	[0.2485, 0.7515]
25000	[0.50052, 0.49948]	[0.25312, 0.74688]
50000	[0.50232, 0.49768]	[0.25122, 0.74878]

Starting position Player 1: [0,1] and Player 2: [1,0]		
Rounds	σ_1	σ_2
10	[0.9, 0.1]	[0.3, 0.7]
50	[0.66, 0.34]	[0.24, 0.76]
100	[0.6, 0.4]	[0.27, 0.73]
250	[0.492, 0.508]	[0.28, 0.72]
500	[0.486, 0.514]	[0.244, 0.756]
1000	[0.527, 0.473]	[0.243, 0.757]
2500	[0.4956, 0.5044]	[0.244, 0.756]
5000	[0.498, 0.502]	[0.244, 0.756]
10000	[0.5085, 0.4915]	[0.2523, 0.7477]
25000	[0.50044, 0.49956]	[0.25392, 0.74608]
50000	[0.49886, 0.50114]	[0.2512, 0.7488]

Starting position Player 1: [0,1] and Player 2: [0,1]		
Rounds	σ_1	σ_2
10	[0.8, 0.2]	[0.4, 0.6]
50	[0.62, 0.38]	[0.3, 0.7]
100	[0.51, 0.49]	[0.32, 0.68]
250	[0.52, 0.48]	[0.22, 0.78]
500	[0.534, 0.466]	[0.238, 0.762]
1000	[0.522, 0.478]	[0.26, 0.74]
2500	[0.518, 0.482]	[0.246, 0.754]
5000	[0.514, 0.486]	[0.2478, 0.7522]
10000	[0.4972, 0.5028]	[0.2523, 0.7477]
25000	[0.50128, 0.49872]	[0.24656, 0.75344]
50000	[0.5024, 0.4976]	[0.2483, 0.7517]

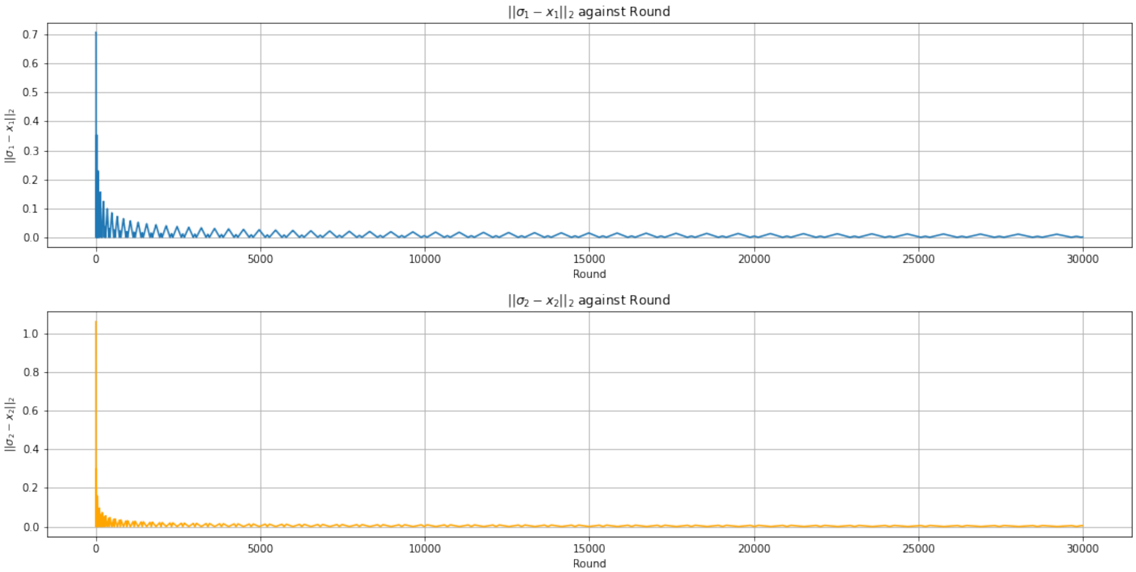
We can see that for all possible starting positions, we have that as the number of rounds increases we have that $\sigma_1 \to x_1 = ((\frac{1}{2}, \frac{1}{2})$ similarly we have that $\sigma_2 \to x_2 = ((\frac{1}{4}, \frac{3}{4})$, where x is the NE of this game given by: (x_1, x_2). I have decided to run simulations for the set of [10,50,100,250,500,1000,2500,5000,10000,25000,50000] rounds, where I am just running the game that is played by both of the players.

However, we can take this analysis one step further. We will show that this is the case graphically. To do this we will make use of properties of norms. We know that $||y||_2$ that is the L2-norm of any vector is given by $||y||_2 = \sqrt{\sum_{i=1}^{N}(y_i)^2}$. We know that if $||y||_2 = 0$ then it must be the case that y is strictly the zero vector that is $y = \tilde{0}$. Therefore, we have that $||y||_2 = 0$ if and only if $y = \tilde{0}$.

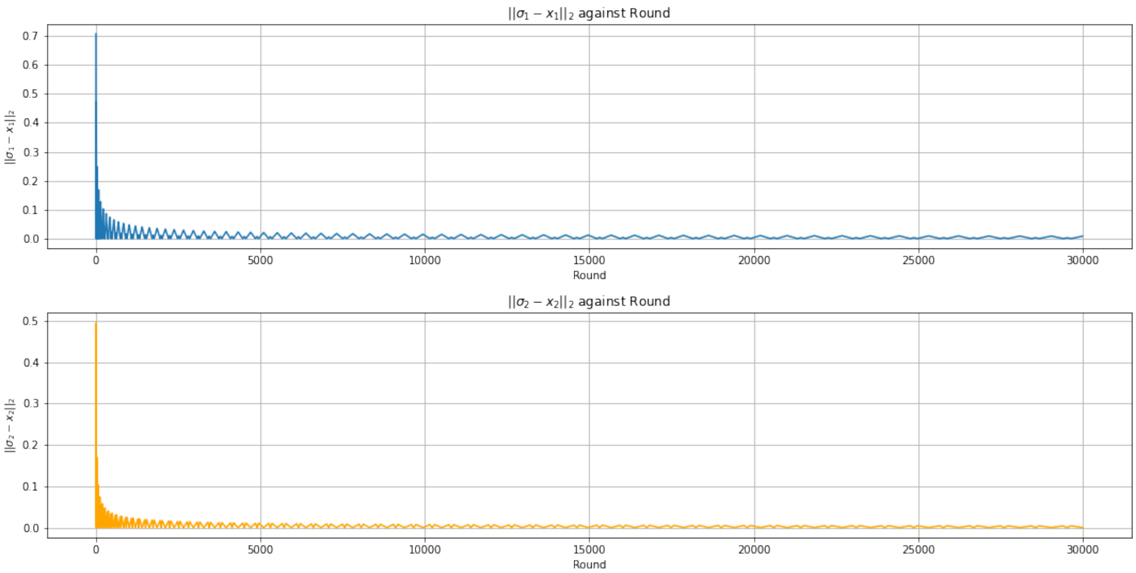
We can use this property of L2-norm, if we can show that if $||\sigma_i - x_i||_2 \to 0$ as the number of rounds tends to infinity we know that σ_i must converge to x_i. This is because $||\sigma_i - x_i||_2 \iff \sigma_i - x_i = \tilde{0}$ so we must have that $\sigma_i - x_i = \tilde{0}$ therefore, $\sigma_i = x_i$ as the number of rounds tends to infinity. Therefore, what I am doing in this case is for every round I calculate σ_1 and σ_2 and then find the L2-norm, that is $||\sigma_1 - x_1||_2$ and $||\sigma_2 - x_2||_2$. I then plot this value against the round. We hope that as the number of rounds increases the L2-norm goes to zero for both σ_1 and σ_2.

We can see that for all of the possible starting positions the L2-norm for both σ_1 and σ_2 go to zero as the number of iterations goes to infinity. That means that these 2 "statistical mixed strategies" do indeed go to the NE x_1 and x_2 for all starting positions. We have therefore shown that this is the case. Note that we can use any tie breaking rule, I have used that in case of a tie Player 1 plays [0,1] and Player 2 plays [1,0]. **Below are the plots for the numerical tests carried out.**

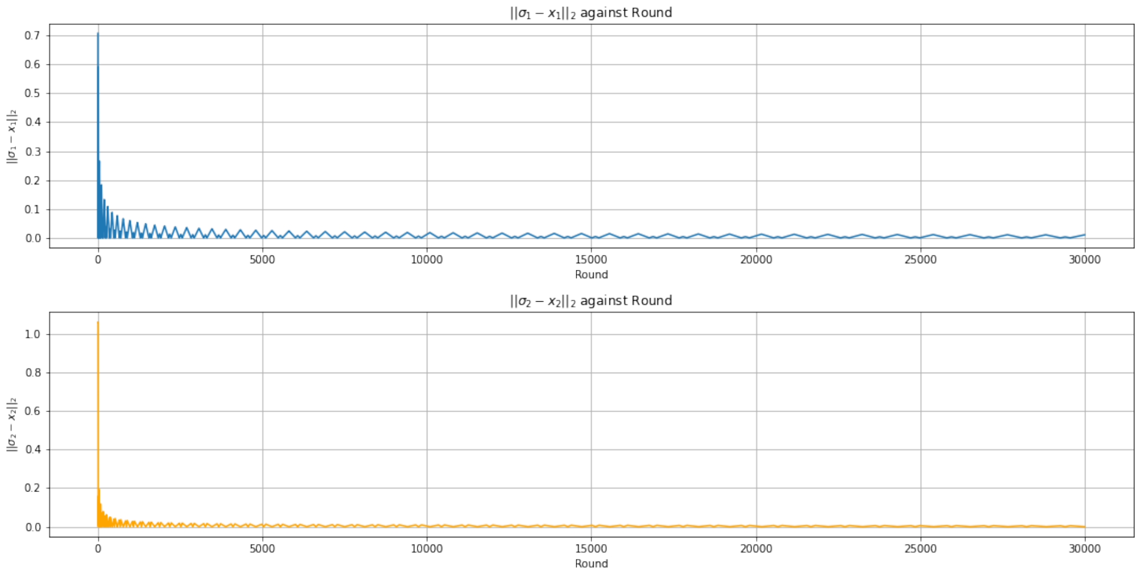
Plotting 2-norm against Round. Strating at pure strategies Player 1 = [1, 0] and Player 2 = [1, 0]



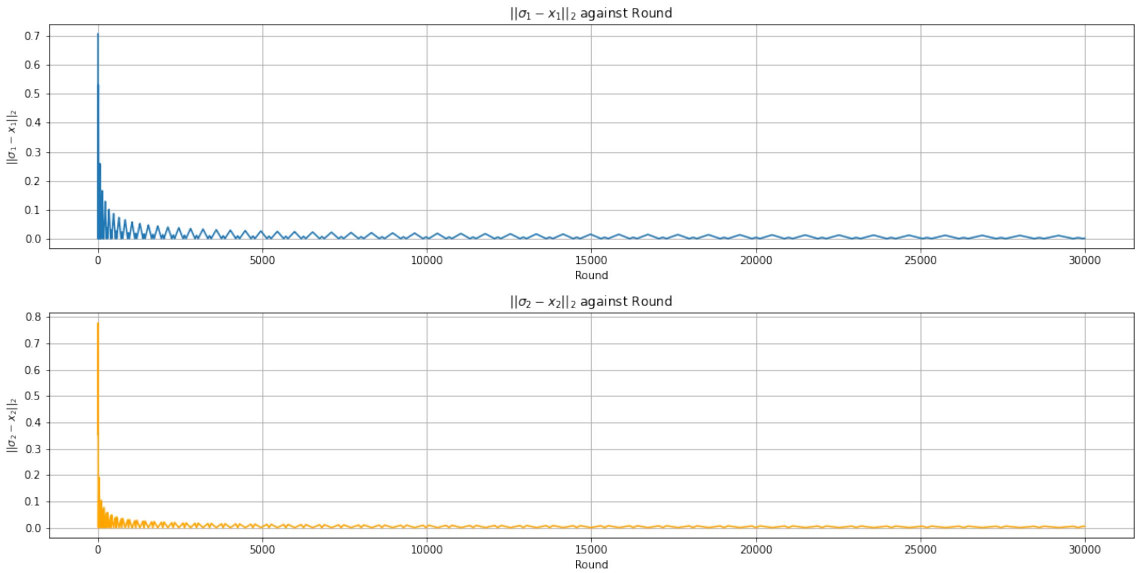
Plotting 2-norm against Round. Strating at pure strategies Player 1 = [1, 0] and Player 2 = [0, 1]



Plotting 2-norm against Round. Strating at pure strategies Player 1 = [0, 1] and Player 2 = [1, 0]



Plotting 2-norm against Round. Strating at pure strategies Player 1 = [0, 1] and Player 2 = [0, 1]



Below I have attached the script used to produce the results. This script here is used to create the game.

```
import numpy as np
import pandas as pd

def expected_payoff_1(sigma,x,A):
    """
    Takes in sigma, x and A and returns the expected payoff for player 1, given by
        (x^T)A(sigma) We use the fact that we know that the expected payoff is equal to
        xAy.

    Input:
    -sigma : 1x2 vector, which is the statistical mixed strategy" player 1 calculates
        (Array)
    -x: either [1,0] or [0,1], one of player 1s pure strategy (Array)
    -A: Payoff matrix (np.matrix)

    Returns:
    -Expected payoff (int)
    """
    return (x@A@sigma).tolist()[0][0]

def expected_payoff_2(sigma,y,A):
    """
    Takes in sigma, x and A and returns the expected payoff for player 2, given by
        (sigma^T)A(y) We use the fact that we know that the expected payoff is equal to
        xAy.

    Input:
    -sigma : 1x2 vector, which is the statistical mixed strategy" player 2 calculates
        (Array)
    -y: either [1,0] or [0,1], one of player 2s pure strategy (Array)
    -A: Payoff matrix (np.matrix)

    Returns:
    -Expected payoff (int)
    """
    return (sigma@A@y).tolist()[0][0]

def pick_strategy(player,sigma, A):
    """
    For a given player it picks their optimal best response for the statistical mixed
        strategy" they calculate.

    Input:
    -player: either 1 or 2 indicating which player is playing (int)
    -sigma: statistical mixed strategy" calculated by player 1 or 2 (Array)
    -A: Payoff matrix (np.matrix)

    Output:
    -optimal best response for player 1 or 2 ([1,0] or [0,1])
```

```
    """
    #These are the possible pure strategies
    pure_strategies = [[1,0],[0,1]]

    if(player == 1):
        PS_1 = expected_payoff_1(sigma,pure_strategies[0],A)
        PS_2 = expected_payoff_1(sigma,pure_strategies[1],A)
        #Player 1 picks strategy which maximises the expected payoff
        if(PS_1 > PS_2):
            return pure_strategies[0]
        else:
             return pure_strategies[1]

    else:
        PS_1 = expected_payoff_2(sigma,pure_strategies[0],A)
        PS_2 = expected_payoff_2(sigma,pure_strategies[1],A)
        #Note that player 2 is trying to minimize the expected payoff so they
        #pick the strategy which minimizes the payoff
        if(PS_1 > PS_2):
            return pure_strategies[1]
        else:
             return pure_strategies[0]

def run(iterations, A, start_1, start_2):
    """
    This function runs plays this strategy update method for both players in repeated play
        for a given number of iterations.

    Input:
    -iterations: number of runs (int)
    -A: Payoff matrix (np.matrix)
    -start_1: player 1s strating strategy ([1,0] or [0,1])
    -start_2: player 2s strating strategy ([1,0] or [0,1])

    Output:
    -list_sigma_1, list_sigma_2: list of all the statistical mixed strategies of the two
        players at each iteration.
    """

    #Variables to count number of heads and tails for each player
    N_1, M_1, N_2, M_2 = 0, 0, 0, 0

    #List of calculated sigmas at each round
    list_sigma_1, list_sigma_2 = [], []

    #Starting strategies for each player
    player_1 = start_1
    player_2 = start_2

    #Increase counter accordingly
    if player_1 == [1,0]:
        N_1+=1
```

```
    else:
        M_1+=1

    if player_2 == [1,0]:
        N_2+=1
    else:
        M_2+=1

    #Calculate sigma_1 and sigma_2
    sigma_1 = [N_1/(N_1+M_1),M_1/(N_1+M_1)]
    sigma_2 = [N_2/(N_2+M_2),M_2/(N_2+M_2)]

    #Loop of the rounds
    for i in range(iterations):

        #Add sigmas to the appropriate list
        list_sigma_1.append(sigma_1)
        list_sigma_2.append(sigma_2)

        #Decide which strategy Player 1 and 2 play
        player_1 = pick_strategy(1,sigma_2, A)
        player_2 = pick_strategy(2,sigma_1, A)

        #Increase counter accordingly
        if player_1 == [1,0]:
            N_1+=1
        else:
            M_1+=1

        if player_2 == [1,0]:
            N_2+=1
        else:
            M_2+=1

        #Update sigma_1 and sigma_2
        sigma_1 = [N_1/(N_1+M_1),M_1/(N_1+M_1)]
        sigma_2 = [N_2/(N_2+M_2),M_2/(N_2+M_2)]

    return(list_sigma_1, list_sigma_2)
```

The following script is used to produce the tables:

```
def table(rounds, A, start_1, start_2):
    sigma_1_list = []
    sigma_2_list = []

    for x in rounds:
        a, b = run(x, A, start_1, start_2)
        #Get the final value of sigma 1 and 2
        sigma_1_list.append(a[-1])
        sigma_2_list.append(b[-1])

    #Create dataframe to store the variables
    data = {'Rounds':rounds,'$\sigma_1$': sigma_1_list, '$\sigma_2$': sigma_2_list}
    data_frame = pd.DataFrame(data=data)
    return data_frame

rounds = [10,50,100,250,500,1000,2500,5000,10000,25000,50000]
A = np.matrix('3 1; 0 2')

#Creating tables for all the different starting pure strategies
df_1 = table(rounds, A, [1,0], [1,0])
df_2 = table(rounds, A, [1,0], [0,1])
df_3 = table(rounds, A, [0,1], [1,0])
df_4 = table(rounds, A, [0,1], [0,1])
```

And finally, this script is used to produce the graphs:

```
def plot_graph(rounds, A, start_1, start_2):
    import matplotlib.pyplot as plt

    rounds = rounds

    NE_1 = [0.5,0.5]
    NE_2 = [0.25,0.75]

    list_sigma_1, list_sigma_2 = run(steps, A, start_1, start_2)

    #Here we calculate the 2L-norm between sigma_i and x_i for all rounds.
    norm_sigma_1 = [np.linalg.norm(np.array(i)-np.array(NE_1)) for i in list_sigma_1]
    norm_sigma_2 = [np.linalg.norm(np.array(i)-np.array(NE_2)) for i in list_sigma_2]

    list_rounds = [i for i in range(rounds)]

    fig, (ax1, ax2) = plt.subplots(2, 1,figsize=(15, 8))

    fig.suptitle('Plotting 2-norm against Round. Strating at pure strategies Player 1 = {}
        and Player 2 = {}'.format(start_1, start_2))

    ax1.plot(list_step, norm_sigma_1)
    ax2.plot(list_step, norm_sigma_2, color="orange")

    ax1.set(title = "$||\sigma_1 - x_1||_2$ against Round")
    ax2.set(title = "$||\sigma_2 - x_2||_2$ against Round")

    ax1.set(xlabel='Round', ylabel='$||\sigma_1 - x_1||_2$')
    ax2.set(xlabel='Round', ylabel='$||\sigma_2 - x_2||_2$')

    ax1.grid()
    ax2.grid()
    fig.tight_layout()

    fig.savefig("graph_for_{}_{}.png".format(start_1, start_2))
    plt.show()

A = np.matrix('3 1; 0 2')

#Plotting graphs for all possible starting positions
plot_graph(30000, A, [1,0], [1,0])
plot_graph(30000, A, [1,0], [0,1])
plot_graph(30000, A, [0,1], [1,0])
plot_graph(30000, A, [0,1], [0,1])
```