Applications of Linear Algebra in Game Theory

Brian Huang

December 6, 2024

1 Introduction

Game theory is a mathematical framework that analyzes the actions that different players of a game take and how those decisions affect one another. Generally, a game refers to a scenario or simulation with particular rules or parameters that define a player's actions. While modern game theory has a deeply mathematical background, it has become prevalent in a wide variety of fields, including Economics, Computer Science, and Biology. From making it easier to model and visualize games to allowing computers to quickly generate strategies, Linear Algebra can be applied to Game Theory in many different ways. This paper will focus on a few select ways that Linear Algebra and Game Theory intersect.

2 Background information

Here are some useful game theory related definitions:

Payoff matrix

Essentially a payoff matrix represents all the possible rewards a game can provide for a player. For example, the payoff matrix of a rock-paper-scissors game with two players would look something like:

$$\begin{array}{cccc} & \text{p} & \text{s} & \text{r} \\ \text{p} & 0 & 1 & -1 \\ \text{s} & -1 & 0 & 1 \\ \text{r} & 1 & -1 & 0 \end{array}$$

The payoff matrix is a 3x3 matrix because each player has 3 strategies to choose from. The ijth entry represents the payoff for the column player, which we can set to player 1. So if player 1 chooses paper, then we would look at the first column to see the payoff would be depending on the strategy player 2 picks.

Traditionally, a payoff matrix contains a tuple which contains the payoffs for both the column player and the row player. However, it is easier to perform calculations when there are two separate matrices that represent the payoffs for each of the players.

Symmetric Games

The above example payoff matrix is also an example of a symmetric game. A game is symmetric when the game treats both players equally in terms of the rewards it provides. You can tell a game is symmetric when its payoff matrix is anti-symmetric.

$$A = -A^T$$

This also means that to calculate the payoff matrix of the row player, we can use $-A^T$.

Zero-Sum Games

A game where the total sum of all players' gain and loss is equal to zero. Whatever one player takes, some other player will lose.

We see that the example version of rock paper scissors is a zero sum game. **Pure Strategy vs Mixed Strategy**

A player uses a pure strategy if they use the same move at each round of the game. A mixed strategy is a method of playing a game where, at each round of the game, the player chooses a move at random so that each is used at a certain probability.

A mixed strategy can be represented as a stochastic vector where all entries in the vector sum to 1. A pure strategy is a stochastic vector where there is only one non-zero entry, and that non-zero entry is equal to 1.

$$m = \begin{bmatrix} 0.75 \\ 0.125 \\ 0.125 \end{bmatrix}, p = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

m is a mixed strategy and p is a pure strategy.

Expected Value

The expected value refers to the expected payoff a player can anticipate on receiving in a game.

To calculate expected value we can multiply the payoff matrix by the probability matrices that represent the strategies of player 1 and player 2.

$$E = S_1^T P S_2$$

S_1, S_2 represent the strategies of Player 1 and Player 2 respectively, and P represents the payoff matrix with S_1 being the column player.

3 Identifying Strategies via Inspection

3.1 Saddle Points

In some games, there is always a strategy that is clearly better than the others. Optimal strategies can be found by looking for saddle points in a payoff matrix. Before a saddle point can be found, we should first define minimax and maximin. The minimax is the strategy that minimizes the player's maximum loss. In a two player game, the minimax is the minimum of the column maxima. The maximin is the strategy that maximizes the player's minimum winnings. In a two player game, the maximin is the maximum of the row minima.

A payoff matrix has a saddle point when the minimax and maximin are of the same value.

Note: If a payoff matrix does not contain saddle points then there are no pure strategy is optimal.

Example:

Suppose we have a game where both player have two strategies to choose from with the corresponding payout matrix, A,

$$\begin{array}{ccc} & s_1 & s_2 \\ s_1 & 3 & 7 \\ s_2 & -1 & 4 \end{array}$$

To find the minimax, we first find the maximums of each column, which are 3 and 7. Then take the minimum, $min(3,7) = 3$.

To find the maximin, we need to find the minimums of each row, which are 3 and -1. Then take the maximum, $max(3,-1) = 3$.

Both the minimax and maximin are 3. So, there is a saddle point at A_{11}. In this game the column player should always select s_1.

3.2 Reduction by Dominance

By carefully examining the rows and columns of a payoff matrix, it is possible to simplify it by looking for dominated rows and columns.

We will employ a similar strategy as minimax and maximin. First we iterate through the rows looking for rows where all entries are less than or equal to another row. We remove this row. This represents maximizing the potential reward. Next we iterate through the columns looking for columns with entries greater than or equal to other columns. We remove these. This represents minimizing the potential loss. To reduce by dominance for an $m * n$, iterate through all rows m times, each time having a pivot row that checks against all other rows. Do the same but for n times for the columns. Repeat this process until it is not possible to reduce anymore.

Suppose, there are two people playing a point based game. After many rounds, both players have found the average number of points they are able to achieve by playing a specific strategy. Let Player 1 be the column player with 3 strategies, and Player 2 be the row player with 5 strategies. The resulting matrix will be:

$$\begin{bmatrix} 0 & -1 & 5 \\ 7 & 5 & 10 \\ 15 & -4 & -7 \\ 5 & 0 & 10 \\ -5 & -10 & 9 \end{bmatrix}$$

Suppose we want to find the best strategy for Player 1. First, iterate through the rows and see if there rows that are dominate the rest:

$$\begin{bmatrix} 7 & 5 & 10 \\ 15 & -4 & -7 \end{bmatrix}$$

All entries of the second row are greater than all entries of the first row since $7 > 0$, $5 > -1$, $10 > 5$. Likewise, the entries of the second row are greater than or equal to the entries of the 4th and 5th row. However, the second row's entries were not all greater than the third row's. Specifically, $7 \ngtr 15$, so we cannot remove it. Next, iterate through the columns.

$$\begin{bmatrix} 5 & 10 \\ -4 & -7 \end{bmatrix}$$

We removed the first column, because it was greater than all entries of the second column. It is possible to repeat this process for rows once more to get the following:

$$\begin{bmatrix} 5 & 10 \end{bmatrix}$$

Finally, we look at the columns.

$$\begin{bmatrix} 5 \end{bmatrix}$$

4 Finding strategies using Linear Programming

Linear Programming is a computational technique that helps find optimal solutions to systems of linear inequalities. To do so, we first generate a payoff matrix based off a given game. Then, determine an appropriate system of inequalities that accurately represents the payoff matrix. We can then add slack variables to convert the system of inequalities into a system of equations. Then, we use the simplex method to solve for an optimal solution.

We can utilize linear algebra for the simplex method by using something called the "revised simplex method", instead of using a tableau, we only store information in matrices.

We need to ensure the following:

1. The objective is maximization (It is possible to turn a minimization problem to a maximization problem by multiplying everything by -1, but it may not always be useful).
2. All entries in the matrix are positive (If they are not, add by a constant k to all entries so that they are).
3. The system of inequalities should be converted into a system of equations using slack variables.

Once this is done, we will have a matrix A which represents the system of equations and the slack variables, a vector c whose entries contain the objective function's coefficients, and another vector b which contains the left hand side of the system of equations. We will also have a vector x which contains the variables so that we will get the linear programming problem of $max\{c^Tx : Ax = b\}$ We will split up A into two smaller matrices: N and B, where N contains the columns of the non-slack variables and B contains the basic, slack variables.

We will also split up and take the transpose of c into c_β^T and c_N^T which represents the basic and non basic entries of c respectively. We will also split up x to get x_N and x_β in the same manner. We now need to select an entering column and leaving columns. We will use E and L to keep track of vector indices. Now, we solve for y^T in: $y^T = c_\beta^T B^{-1}$.

Then, using a version of Bland's rule, we select a vector by choosing the smallest improving index of z_N^T where $z_N^T = y^T N - c_N^T$, and using the corresponding vector. This is the entering column and will be denoted as a_E, with coefficient x_E.

Now we find the leaving vector. We introduce d where $d = B^{-1}a_E$, and we look for some t so that an entry in $x_\beta(t) = x_\beta(0) - td$ becomes 0.

With this entry, we update our matrices by swapping the E and L columns and the variables corresponding to them, and update our solution by updating a BFS (Basic Feasible Solution) vector. This is more clear with an example.

Continue repeating the process of selecting entering and leaving columns and updating until there are no negative entries in z_N^T.

Example:

Let's define an arbitrary two player game with the following payoff matrix:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Using our previous strategy of checking for saddle points, we see that there are none. So the best strategy will be mixed.

Let p be the stochastic vector with entries p_1, p_2, p_3 which corresponds to Player A's mixed strategy.

Let q be the stochastic vector with entries q_1, q_2, q_3 which corresponds to Player B's mixed strategy.

$$\begin{array}{ccccc} & & \text{Player B} & & \text{prob} \\ & 0 & 1 & 0 & p_1 \\ \text{Player A} & 0 & -1 & 1 & p_2 \\ & 1 & 1 & 0 & p_3 \\ \text{prob} & q_1 & q_2 & q_3 & \end{array}$$

To ensure all entries are non-negative we add to each entry $k = 1$.

$$\begin{array}{ccccc} & & \text{Player B} & & \text{prob} \\ & 1 & 2 & 1 & p_1 \\ \text{Player A} & 1 & 0 & 2 & p_2 \\ & 2 & 2 & 1 & p_3 \\ \text{prob} & q_1 & q_2 & q_3 & \end{array}$$

Now, we can convert to a system of linear inequalities. Let V be the game payoff. Let's say Player A is trying to maximize their expected gains, and Player B is trying to minimize their losses.

Player A	Player B
Maximize $V = p_1 + p_2 + p_3$	Minimize $V = q_1 + q_2 + q_3$
Subject to	Subject to
$p_1 + 2p_2 + p_3 \geq V$	$q_1 + q_2 + 2q_3 \leq V$
$p_1 + 0p_2 + 2p_3 \geq V$	$2q_1 + 0q_2 + 2q_3 \leq V$
$2p_1 + 2p_2 + p_3 \geq V$	$q_1 + 2q_2 + q_3 \leq V$

Now, divide both sides by V, and introduce new variables.

$x_i = \frac{p_i}{V}$ and $y_i = \frac{q_i}{V}$, $1 \leq i \leq 3$. Because as V increases, V decreases, the maximization problem becomes a minimization problem and the minimization problem becomes a maximization problem. We let $\frac{1}{V} = Z$ This results in:

Player A	Player B
Minimize $Z = x_1 + x_2 + x_3$	Maximize $Z = y_1 + y_2 + y_3$
Subject to	Subject to
$x_1 + 2x_2 + x_3 \geq 1$	$y_1 + y_2 + 2y_3 \leq 1$
$x_1 + 0x_2 + 2x_3 \geq 1$	$2y_1 + 0y_2 + 2y_3 \leq 1$
$2x_1 + 2x_2 + x_3 \geq 1$	$y_1 + 2y_2 + y_3 \leq 1$

We will focus on Player B for now, since it is already a maximization problem

To fully utilize these inequalities, we need to add slack variables to turn them into equations. .

$$\text{Maximize } Z = y_1 + y_2 + y_3$$

$$\text{Subject to}$$

$$y_1 + y_2 + 2y_3 + s_1 = 1$$

$$2y_1 + 0y_2 + 2y_3 + s_2 = 1$$

$$y_1 + 2y_2 + y_3 + s_3 = 1$$

Now we convert to matrices and vectors. As stated above, we need a matrix A, a vector c, and a vector x (I called it x although we are looking for variables y_n).

$$\text{A}=\begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 2 & 0 & 2 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & 0 & 1 \end{bmatrix}, \text{c} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{x}= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}, \text{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Now split the matrix and vectors into smaller matrices and vectors.

$$\text{N}=\begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \text{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{c}_N^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, c_\beta^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, x_N = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, x_\beta = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

First find y^T by performing the following:

$$y^T = c_\beta^T B^{-1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Now to get the entering column, we find $z_N^T = y^T N - c_N^T$.

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \end{array} = \text{z}_N^T$$

Using Bland's rule, we will select the first entry which represents x_1, so $E = 1$, $a_E = a_1$.

Now, we select the leaving column. Recall, $d = B^{-1}a_E$, $x_\beta^* = B^{-1}b$, and $x_\beta(t) = x_\beta(0) - td$.

$$x_\beta(t) = \begin{array}{c} s_4 \\ s_5 \\ s_6 \end{array} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \text{t} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{array}{c} s_4 \\ s_5 \\ s_6 \end{array} \begin{pmatrix} 1 - t \\ 1 - 2t \\ 1 - 1t \end{pmatrix}$$

Now we want to look for the smallest t so that an entry in $x_\beta(t)$ is 0. We see that $\frac{1}{2}$ works, we can call this $t*$. So we set $L = 5$. Now we introduce a basic feasible solution (BFS) vector:

$$x^*(t^*) = \begin{bmatrix} x_N(t*) \\ x_\beta(t*) \end{bmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

$x_N(t*)$ starts as $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, here we replace the first entry, because $E = 1$, with $t*$. Now we swap the E column with the L column in N, B and c_N^T, c_β^T, and here we will restate our updated information.

$$N_2 = \begin{array}{c} \begin{array}{ccc} y_2 & y_3 & s_5 \end{array} \\ \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 1 \\ 2 & 1 & 0 \end{bmatrix} \end{array}, B_2 = \begin{array}{c} \begin{array}{ccc} y_1 & s_5 & s_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{array}, c_N^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, c_\beta^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Find y^T by performing the following:

$$y^T = c_\beta^T B^{-1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ 1 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & 0 \end{bmatrix}$$

Now to get the entering column, we find $z_N^T = y^T N - c_N^T$.

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} = \begin{array}{c} \begin{array}{ccc} y_2 & y_3 & s_5 \end{array} \\ \begin{bmatrix} -1 & 0 & \frac{1}{2} \end{bmatrix} \end{array}$$

There is only one negative entry here, so we pick $E = 2$,$a_E = a_2$. We now have an entering column. Now we find a leaving column.

$$d = B_2^{-1}a = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, x_\beta(t) = \begin{array}{c} y_1 \\ s_4 \\ s_6 \end{array} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} - t \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} = \begin{array}{c} y_1 \\ s_4 \\ s_6 \end{array} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} - t \\ \frac{1}{2} - 2t \end{bmatrix}$$

We see that when $t = \frac{1}{4}$ the last entry becomes 0. So the leaving column will be $L = 6$. Now we update our BFS.

$$x^*(t^*) = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \\ 0 \\ \frac{1}{2} \\ \frac{1}{4} \\ 0 \end{pmatrix}$$

Swap the E column with the L column.

$$N_3 = \begin{array}{ccc} y_3 & s_5 & s_6 \\ \hline 2 & 1 & 0 \\ 2 & 0 & 0 \\ 1 & 0 & 1 \end{array}, B_3 = \begin{array}{ccc} y_1 & y_2 & s_4 \\ \hline 1 & 1 & 1 \\ 2 & 0 & 0 \\ 1 & 2 & 0 \end{array}, c_N^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, c_\beta^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

Now we repeat the first step of finding y^T and z_N^T.

$$y^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} B^{-1} = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

This time when we compute z_N^T, all entries will be nonnegative which means we are done.

$$z_N^T = y^T N - c_N^T = \begin{bmatrix} 0 & 0 & \frac{1}{2} \end{bmatrix}$$

To interpret our results we refer back to the basic feasible solution vector.

$$x^*(t^*) = \begin{array}{c} y_1 \\ y_2 \\ y_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \\ 0 \\ \frac{1}{2} \\ \frac{1}{4} \\ 0 \end{pmatrix}$$

The optimal solution can be stated as:

$y_1 = \frac{1}{2}, y_2 = \frac{1}{4}, y_3 = 0, \text{Max } Z = \frac{3}{4}$

We found Z by adding the values of the slack variables.

$Z = \frac{3}{4} = \frac{1}{V}, V = \frac{4}{3}$

We multiply each y by V to find the mixed strategy that minimizes Player B's loss.

$$(\frac{1}{2} * \frac{4}{3}, \frac{1}{4} * \frac{4}{3}, 0) = (\frac{2}{3}, \frac{1}{3}, 0)$$

This was a fairly simple example, and was fairly simple to compute by hand. However, the point of the revised simplex method is to make things computationally efficient when implementing it in computer programs. One thing to note is the possibility of degeneracy. This issue can be solved by adding in checkers to see if improvements are being made every iteration. If not, then prematurely ending the program would be good.

5 Stochastic Games

Stochastic Games (or Markov Games) are a particular subset of games where the environment is capable of continuously changing which can affect the outcomes of players' decisions. These types of games allow for flexibility that may not be present in others. At the bare minimum a 2 player Stochastic Game with n states will contain the following:

A $n \times n$ transition matrix that describes how to current environment and players' actions affects the state of the next round.

A $n \times 1$ vector with the entries denoting the probabilities of starting at a specific state.

n separate $j \times k$ payoff matrices representing the rewards given by choosing different strategies, where j represents the strategies of the row player and k represents the number of strategies the column player has.

Example

We can create a modified version of the prisoner's dilemma. This is a two player game, with following payoff matrix:

$$\begin{array}{ccc} & C & D \\ C & b-c & -c \\ D & b & 0 \end{array}$$

A player can either Cooperate or Defect. We now introduce two payoff matrices X, Y with $b_1 = 2, c_1 = 1$ and $b_2 = 1.2, c_2 = 1$.

$$\text{X}=\begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix} \qquad \text{Y}=\begin{bmatrix} 0.2 & -1 \\ 1.2 & 0 \end{bmatrix}$$

Say the starting state vector is $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ so the game will always start in state X. The transitions from state to state will be based off of whether or not the players cooperate. The following depicts a graphical representation of the stochastic game.

$$\begin{tikzcd} X \arrow[loop left, "2C"] \arrow[r, bend left, "\text{0C,1C}"] & Y \arrow[l, bend left, "2C"] \arrow[loop right, "\text{0C,1C}"] \end{tikzcd}$$

C in this context represents the number of players that select "Cooperate".

Now we define 3 different strategies called cooperate, medium, defect, which are all stochastic vectors equal to:

$$c = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, m = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, d = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

These represent some sample strategies the two players can take. I coded a random walker in matlab which pitted the different strategies against one another. The results were the following:

	C	M	D
C	0.73	0.92	1.04
M	-0.12	0.18	0.55
D	-0.79	-0.39	0.01

The entries represent the time the column player spends in jail based on the strategy the row player selected. Even though, I used static strategies, I think implementing dynamic strategies that actually adapt to the environment would probably provide more interesting results.

6 Concluding Notes

Throughout this paper, we have identified 3 main uses of linear algebra in game theory. By converting a given game into matrices it is possible to quickly reduce the complexity of the game, or even find an optimal strategy just from inspection. Utilizing linear programming can help identify more complex strategies. In particular, the revised simplex method makes it easy to implement on computers. Lastly, we introduced Stochastic Games which use Markov Chains to keep track of the different states a game can be in.
For the most part, we focused on games with 2 players. This is because a payoff matrix can only represent the strategies of two players. However, more complex games involving more players can be represented using tensors. Tensors can be said to be generalizations of vectors and matrices in higher dimensions. A 1 dimensional tensor would be a vector, while a 2 dimensional tensor would be a matrix.

7 References

https://www.cs.princeton.edu/courses/archive/spring15/cos511/notes/lec_11_kiran.pdf

https://www.zweigmedia.com/RealWorld/Summary3b.htmlrd

https://arxiv.org/pdf/1404.0086

https://pmc.ncbi.nlm.nih.gov/articles/PMC4653174/

https://digitalcommons.morris.umn.edu/cgi/viewcontent.cgi?article=1004&context=capstone

https://arxiv.org/pdf/1810.08798

https://www.matem.unam.mx/ omar/math340/matrix-games.html

https://personal.math.ubc.ca/ loew/m340/rsm-notes.pdf

https://cbom.atozmath.com/example/CBOM/GameTheory.aspx?q=lppq1=E2

https://www.pnas.org/doi/10.1073/pnas.1908643116sec-3

https://static-content.springer.com/esm/art%3A10.1038%2Fs41586-018-0277-x/MediaObjects/41
586_2018_277_MOESM1_ESM.pdf

https://www.nature.com/articles/s41586-018-0277-x

8 Code

This is code for the random walker.

```
runs = 1

samp = 1000

%var defs

X = [1 -1;2 0];

Y = [0.2 -1; 1.2 0];

c = [0.9;0.1];

m = [0.5; 0.5];

d = [0.1;0.9];


total = 0;

aggregate = 0;

avg =0;


%start in matrix X

curr_mat = X


%set player strategies

p1_str = d

p2_str = m

for i = 1:runs

for j = 1:samp

    coop_count = 0;


    %eval p1

    r1 = rand();

    if r1<p1_str(1:1)

        coop_count=coop_count+1;

        p1_select = 1;

    else

        p1_select=2;

    end

    %eval p2
    r2 = rand();
    if r2<p2_str(1:1)
        coop_count=coop_count+1;
        p2_select = 1;
    else
        p2_select=2;
    end
%update totals
    %sum = transpose(p1_str)*curr_mat*p2_str %take ev
    sum=curr_mat(p1_select,p2_select);
    total=total+sum;



%swap curr_max if necessary
if (isequal(curr_mat,X))
    if(coop_count==2)


    else
        curr_mat=Y;
    end
end
if (isequal(curr_mat,Y))
    if(coop_count==2)
        curr_mat=X;
    end
end


end
%nvm
aggregate = aggregate+(total/samp);


end
finale = aggregate/runs
```