Derreck Ross
Math 480

Computational Economics: Finding Pure and Mixed Strategy Nash Equilibrium with Sage Math

Abstract:

Game theory has become an important part of economics within the last century. The study of game theory has been used to explain the occurrence of certain social and economic phenomena. This paper seeks to utilize the mathematical power of Sage Math, and add functionality to solve economic games. The project outlines the procedures used to implement a game solver, and shows the code's ability to solve normal games for pure and mixed strategy Nash Equilibrium.

Background information:

One of the early concepts taught in economic game theory is the Nash Equilibrium. The term is based off of John F. Nash Jr's definition which was published in 1950, although its ideology has been around much longer. Nash's definition from "Equilibrium Points in N-Person Games" says that given an n-tuple of strategies for each player, they are in equilibrium if every strategy in the tuple "counters" the other strategies in the tuple, where "counters" is defined as yielding the highest obtainable expectation for each player given the other N – 1 strategies of the other players (Nash 1950).

A more simple way to describe Nash Equilibrium is to say that it is a set of strategies in which every player's strategy is the "best response" to each other player's strategy. This is equivalent to saying each player cannot personally gain by changing their strategy alone. The concept of a Nash Equilibrium helps explain some of the counter-intuitive behavior seen in social and economic games.

Perhaps the most popular example of explaining the Nash Equilibrium is the Prisoner's Dilemma. The Prisoner's Dilemma is a fictional situation where two bank robbers have been

caught and are being interrogated separately by the police. They are both told that they have the option to remain silent or fink on their partner. If both players remain silent, the police do not have enough information to convict them for the bank robbery, so they both serve 1 year in prison for a different minor charge. If one player finks and the other remains silent, then the one who remained silent will serve 10 years in prison, while the player who finked gets no prison sentence. Lastly, if both players fink on each other, they both serve 7 years in prison for the bank robbery. Even knowing the outcomes of their choices, both prisoners will fink on each other, despite that being a much worse outcome than both staying silent. Their reasoning is based on a Nash Equilibrium. If we write the outcomes of this game in a table, where the first number represents the payoff of prisoner 1 and the second number represents the payoff of player 2, we can see the problem more clearly:

		Prisoner 2	
		Remain Silent	Fink
Prisoner 1	Remain Silent	-1,-1	-10, 0
	Fink	0, -10	-7, -7

Using the definition of Nash Equilibrium, we can see that the only point where neither player will gain by deviating alone is the point (Fink, Fink) which is marked in yellow. This means that at any other point in the table, at least one of the players can gain by changing their answer to fink.

Nash stated in his paper that for every game with a finite number of players and a finite number of pure strategies, there is at least one Nash Equilibrium (Nash 1950). For games where a player could gain by deviating at all points within the table, the concept of mixed strategies

must be used. Take for example the game of matching pennies. In this game, player 1 sets a penny to heads or tails and hides the answer from his opponent, and player 2 does the same. After making their decisions, both players reveal their coins at the same time. If the side of both coins matches: (heads, heads) or (tails, tails), player 1 receives 1 dollar from player 2. If the sides of both coins do not match: (heads, tails) or (tails, heads), player 2 receives 1 dollar from player 1. This is shown in the following rewards table:

		Player 2	
		Heads	Tails
Player 1	Heads	1,-1	-1,1
	Tails	-1,1	1,-1

Unlike the Prisoner's Dilemma, a player could gain by deviation at all points in the table. The Nash equilibrium must be a mixed strategy in this case. A mixed strategy occurs when a player plays a combination of pure strategies with a specific probability for each pure strategy. In the matching pennies game, the mixed strategy Nash Equilibrium is p0 = 0.5 and q0 = 0.5 where p0 is the probability that player 1 should play heads and q0 is the probability that player 2 should play heads. This strategy makes the other player indifferent between the choice of heads or tails.

Program implementation:

The first goal of this project was to find the pure strategy Nash Equilibrium (or Equilibria) from a payoff matrix. To do this, the code takes two numpy arrays. Numpy was chosen for its efficient searching and manipulation of arrays. The arrays are designed as the payoffs of each outcome for each player. In the example regarding the Prisoner's Dilemma above, the numpy arrays are:

```
A1 = np.array([[-1,-10],[0,-7]])
print A1
A2 = np.array([[-1,0],[-10,-7]])
print A2
```

```
[[ -1 -10]
 [  0  -7]]
[[ -1   0]
 [-10  -7]]
```

Where the top left entry is -1 in both arrays because the payoff for both players in the top left part of the payoff matrix.

A method to find the pure strategy Nash Equilibrium is described in Robert Gibbons' book *Game Theory for Applied Economists*. He outlines a simple yet effective way to find the pure strategy Nash Equilibrium for any game in normal form, which is where the payoffs are written in a matrix as we have seen in the background information. His method involves taking the best response to the other player's options by marking the largest payoff for player 1 in each column and the largest payoff for player 2 in each row. When a matrix cell has both payoffs marked, it is a pure strategy Nash Equilibrium, as the choices are best responses for all players (Gibbons 1992).

This was implemented in the code by using the max() function to find the largest value in each column for player 1 and each row for player 2. The code then adds the coordinates of the payoff if it matches the max value. This method is superior to using the argmax() function in numpy because there could be multiple payoffs that have the max value, and the argmax() function only finds the first one.

```
for j in range(columns):
    high1_val = player_1[:,j].max()
    for i in range(rows):
        if player_1[i,j] == high1_val:
            high1_index.append([i,j])

for i in range(rows):
    high2_val = player_2[i,:].max()
    for j in range(columns):
        if player_2[i,j] == high2_val:
            high2_index.append([i,j])
```

Then the results of for each player are then iterated. If both lists contain the same coordinates, the matching coordinates relate to the pure strategy Nash Equilibrium.

For mixed strategy, the method taught by Jacques Lawarree at the University of Washington was used. His method involved setting the players payoff for each choice equal to each other, and thereby finding the other players probability of playing each of their options. To give an example using matching pennies, if we allow q0 to be the probability of player 2 playing heads and q1 to the probability of player 2 playing tails, the payoff for player 1 should be:

Payoff (Heads) = 1*q0 + (-1)*q1

Because when player 1 plays heads, he receives a payoff of 1 when player 2 plays heads, and a payoff of -1 when player 2 plays tails (Lawaree 2011).

When the payoffs of all choices a player can make are equal to each other, one can solve for q0 and q1 with the extra requirement that the sum of all q's should be equal to 1. Solving n equations for n unknowns is very simple with the solve() function in Sage:

```
mix_NE = []
mix_NE += solve([equalities1, sum(vars2) == 1],vars2)
mix_NE += solve([equalities2, sum(vars1) == 1],vars1)
```

Where equalities is a list of all the equations that must be equal to each other, and vars is the p and q values represent the probabilities of each choice. Notice that the equalities for player 1 solve the probabilities for player 2. As stated above, setting payoffs equal for one player solves the other's mixed strategy choices.

Testing analysis:

As shown in the Sage worksheet which is published here: Game Theory

Example A solves the prisoner's dilemma. As we can see the option (Fink, Fink) is the pure strategy Nash Equilibrium.

Example B solves matching pennies. As we can see, there is no pure strategy Nash Equilibrium, but rather a mixed strategy Nash Equilibrium where each player plays heads or tails with a probability of 0.5.

Example C was a midterm question asked in Econ 485. The solution was indeed (3,2).

Example D shows the solution to a random 1000 by 1000 matrix for both players. While it would be unlikely for both players to have 1000 choices, this example just shows how quickly the pure strategy Nash Equilibrium solver is. As you can see, it analyzes 2 million pieces of data and returns a Nash Equilibrium in about 1 second.

Results:

The program was a resounding success. As we can see from the samples, the code can solve most basic problems in economic game theory.

Improvements:

There is much room for improvement in this code. Some of the ideas that could be implemented if there was more time:

- An option for more than 2 players. This would be very difficult to do with the mixed strategy equilibrium, as there would have to be new unique variables for each player's probability of choosing an option.
- Adding functionality to solve iterated games, which are games where players take turns instead of acting simultaneously.
- Solving games of incomplete information where some players are not aware of some attribute within the game.

There are many others as well, but this code is a good start into solving some of the very basic structures in game theory.

Conclusions:

Using the coding tools that I learned in Math 480 and the Economic Game Theory I learned from Econ 485, I have created a program that quickly solves the pure and mixed strategy Nash Equilibrium for two person games. While there is still much to do terms of computational economics, this is a good start. The main purpose of this program was for educational purposes, but I would like to someday see this type of code used to quickly solve simple game theory problems.

References:

Gibbons, Robert. *Game theory for applied economists*. 1st ed. Princeton: Princeton Univ Pr, 1992. 8-11. Print.

Lawarree, Jacques. "Mixed Strategy." Economic Game Theory. UW. Savery Hall, Seattle, WA. Spring 2011. Lecture.

Nash, John F. "Equilibrium Points in N-Person Games." *Proceedings of the National Academy of Sciences of the United States of America* 36.1 (1950): 48-49. Web. 2 Jun 2011. <http://courses.engr.illinois.edu.offcampus.lib.washington.edu/ece586/TB/Nash-NAS-1950.pdf>