2022

Implementing the Simplex Algorithm to Solve Zero-Sum Games

Chineng Vang
University of Minnesota - Morris, vang2660@morris.umn.edu

IMPLEMENTING THE SIMPLEX ALGORITHM TO SOLVE ZERO-SUM GAMES

CHINENG "COOKIE" VANG
MATH 4901 - SENIOR SEMINAR
FACULTY ADVISOR: DAVID ROBERTS
SECOND READER: CHRIS ATKINSON
UNIVERSITY OF MINNESOTA, MORRIS

Abstract. The concept of a zero-sum game in Game Theory is widely studied in fields such as Mathematics and Economics in order to analyze situations and determine optimal courses of actions. Although a game or situation with fewer decisions can be solved by hand, we can turn to the Simplex Algorithm to efficiently solve larger games. This paper examines how the Simplex Algorithm can be implemented as a linear program in order to solve $m \times n$ zero-sum games.

1. Introduction

Decision making is a process people perform daily albeit 90% of these decisions are made unconsciously [2]. Decisions that hold significant consequence often require greater rationale and effort. Game Theory, the study of strategic decision making, is a prime example of this idea.

John von Neumann originally introduced Game Theory in his 1928 paper *On the Theory of Games of Strategy*. Neumann and Oskar Morgenstern then expanded upon it with their book *Theory of Games and Economic Behavior* in 1944 [3]. A key topic of interest was two-person zero-sum games, games played by two people where the gains or winnings of one player are directly attributed to the losses of the other player. A central assumption is that players will act rationally, meaning they will always follow what is mathematically deemed as the "best" strategy. This paper will refer to the two players in a game as Alice (she/her) and Bob (he/him).

In this paper, we will examine two-person zero-sum games to discover the best strategy for each player as well as the outcome when those strategies are used. In general, a strategy is a course of action or decisions available to a player during a game. Games can be represented

Date: Spring 2022.

by a matrix, which holds the strategies available to both players along with the outcome of a game when players play each of their strategies. Games represented by small matrices are first introduced along with a few methods to solve them by hand. Then we introduce the necessary tools to make the Simplex Algorithm work, and finally discuss the Simplex Algorithm along with its implementation in Mathematica.

1.1. **Definitions and Context.** To define the relevant terms and concepts, let us begin with a game called Swords. Suppose on the count of three Alice and Bob can each choose to show the other person either 1 finger or 2 fingers. If the sum of the shown fingers is even, Bob pays Alice 1 dollar. If the sum of the shown fingers is odd, Alice pays Bob 1 dollar. This game can be represented in the following matrix:

Alice\Bob	1 Finger	2 Fingers
1 Finger	1	-1
2 Fingers	-1	1

TABLE 1. The payoff matrix for Swords. Alice is the row player and Bob is the column player.

Alice and Bob each have two ways to play the game called **strategies**. They can either show 1 or 2 fingers. We call Alice the **row player**, because her strategies are along the left side of the matrix. Bob is the **column player** because his strategies are along the top of the matrix. The numbers in the matrix are called **payoffs** (and we will refer to the matrix as a payoff matrix). Payoffs represent the outcome of a game based on the strategy chosen by each player. We will choose to think of the payoffs as Alice's gains and Bob's losses. To demonstrate, if Alice chooses the strategy to show 1 finger, and Bob chooses the strategy to show 1 finger, then the result is an even number and the corresponding payoff of the game is 1. This means Alice gains 1 dollar from Bob or dually that Bob loses 1 dollar to Alice. If Bob had chosen to show 2 fingers and Alice 1 finger, then Alice gains -1 dollar and Bob loses -1 dollar. This is a convention nuance for simplicity in the Simplex Algorithm as we could easily represent the gains and losses of the players in different ways.

A question that naturally follows is how should Alice and Bob play each of their available strategies so that both players are playing optimally? That is, Alice is choosing a strategy ensuring she gains as much

as possible, and Bob chooses a strategy ensuring he loses as little as possible. This solution for each player is called an **optimal strategy**. Recall that we are denoting the payoffs in the matrix to be Alice's gains and simultaneously Bob's losses aligning with the principle of a zero-sum game. The optimal strategy for the row player Alice guarantees a floor on her gains. By playing her optimal strategy, she will always gain at least the floor value, but can potentially gain more based on how Bob plays. In a similar manner, Bob's optimal strategy guarantees a ceiling on his losses. He can lose at most the ceiling value when playing his optimal strategy, but can potentially lose less based on how Alice plays. A player's optimal strategy can either be a pure strategy or a mixed strategy. A **pure strategy** is when a player only ever plays one of their available strategies. A **mixed strategy** is when a player plays more than one strategy. Regardless of whether a strategy is pure or mixed, it can be represented as a vector of probabilities that add up to 1. For the example in Table 1, the optimal strategy for each player is to use a mixed strategy, playing their two strategies half the time. Therefore the optimal strategy for Alice and Bob is $\langle \frac{1}{2}, \frac{1}{2} \rangle$.

A **solution** to a payoff matrix is a triple consisting of the optimal strategy of the row player, the optimal strategy of the column player, and the game value, denoted (A_i, B_j, V). The **game value** is simultaneously the floor of Alice's gains and the ceiling of Bob's losses. If Alice plays her optimal strategy, this is the lowest she can guarantee to gain with the potential to gain more. If Bob plays his optimal strategy, this is the most he can expect to lose with the potential to lose less.

2. Solving Smaller Matrices By Hand

2.1. **Saddle Point Solutions.** Consider the payoff matrix in Table 2. Our goal is to find a solution to the payoff matrix, so optimal strategies for Alice and Bob, as well as the game value. The simplest way to determine the game value of the payoff matrix is by determining if an entry in the payoff matrix is simultaneously the lowest value in its row and the largest value in its column, known as a **saddle point**. If there is a saddle point, the game has what is called a **Saddle Point Solution**:

Definition. Let the entry in the ith row and jth column of a game's payoff matrix be denoted a_{ij}. Suppose there are pure strategies A_h and B_k in a payoff matrix where the entry a_{hk} is both a least element in the row A_h and also the greatest element in the column B_k. Then

the triple (A_h,B_k,a_{hk}) is a solution of the game, where A_h and B_k are optimal strategies and a_{hk} is the game value.

Alice\Bob	I	II
I	3	7
II	−1	4

TABLE 2. A sample 2×2 payoff matrix.

To determine if a payoff matrix has a saddle point, we need to find what is known as the maximin and minimax. The **maximin** is the largest value out of the smallest values of each row. The **minimax** is the smallest value out of the largest values of each column. If the maximin and minimax values are the same, then the entry in the matrix that leads to both is a saddle point.

Alice\Bob	I	II	Row minimums
I	3	7	3
II	−1	4	−1
Column maximums	3	7	Game Value = 3

TABLE 3. The minimax (cyan), maximin (orange), and game value (yellow).

In this example, we find that the value 3 is the maximin and minimax. Therefore the entry in spot a_{11} of the matrix is a saddle point and the game value since it is in the row of the maximin and column of the minimax. By the definition of a saddle point solution, a solution is $(A_1,B_1,3)$. Intuitively, this makes sense for Alice as choosing her first strategy will always lead to a more favorable outcome than her second strategy. This is same case for Bob. Finding a Saddle Point Solution conveniently applies to any size game matrix although it is more probable to occur in smaller payoff matrices.

2.2. **Algorithm to Solve a 2×2 Game.** If there are no saddle points, then for 2×2 games we can turn to an algorithm which derives formulas to solve for a solution of the matrix.

		q	$1-q$	
	Alice\Bob	I	II	Bob's losses
p	I	a	b	$aq+b(1-q)$
$1-p$	II	c	d	$cq+d(1-q)$
	Alice's gains	$ap+c(1-p)$	$bp+d(1-p)$	

TABLE 4. The probabilities that Alice plays her two strategies are p and 1-p. For Bob, they are q and 1-q.

To begin, consider the payoff matrix in Table 4. We denote the probabilities that Alice plays her strategies in a vector as $\langle p, 1-p\rangle$. For Bob, we denote them as $\langle q, 1-q\rangle$. Alice's gains are along the bottom of the payoff matrix and Bob's losses are along the right side of the payoff matrix. These gains and losses are called **marginal values** representing what Alice and Bob can expect to gain and lose (respectively) based on the the other player's strategies. As an example, the marginal value $ap+c(1-p)$ is what Alice can expect to gain against Bob's first strategy. It is found by taking the dot product of Alice's strategies $\langle p, 1-p\rangle$ and the values in the first column written as a vector $\langle a, c\rangle$.

We can use these marginal values to solve for p, $1-p$, q, and $1-q$. Recall that Alice is trying to find an optimal strategy, and this was done by finding a floor on her gains. We did this in the previous section by finding the maximin of the values along the bottom of the matrix. Alice clearly wants to find the largest floor possible. This can be accomplished by setting the two marginal values equal to each other, as the marginal values along the bottom depend on p. The floor would have to be the smaller of the marginal values when p is plugged in. This encourages us to set the two marginal values equal, so that the floor can be as large as possible:

$$ap+c(1-p)=bp+d(1-p)$$

With some algebra, we can solve for p in terms of a, b, c, and d:

$$p=\frac{d-c}{a-b-c+d}$$

The value $1-p$ now comes easily:

$$1 - p = 1 - \frac{d - c}{a - b - c + d}$$

$$1 - p = \frac{a - b}{a - b - c + d}$$

So now we have the probabilities p and $1-p$ with which Alice should play her two strategies based on the entries of the matrix. A similar process is used to find q and $1 - q$ by using the marginal values on the right side of the matrix:

$$aq + b(1 - q) = cq + d(1 - q) \tag{1}$$

$$q = \frac{d - b}{a - b - c + d}$$

$$1 - q = \frac{a - c}{a - b - c + d} \tag{2}$$

For Bob, the motivation for setting the values equal to each other in equation 1 is swapped from Alice's as he wants the ceiling on his losses to be as small as possible. One point of concern is that the equations for p, $1 - p$, q, and $1 - q$ only hold if a-b-c+$d \neq 0$. If it turns out that a-b-c+$d = 0$, then we can fortunately see that the matrix has a saddle point and use the technique in the previous section.

Seen in Table 5, we can now fill in p, $1 - p$, q, and $1 - q$ into the original payoff matrix, as well as recalculate the marginal values. We currently have the optimal strategy for Alice (p and $1 - p$) and Bob (q and $1 - q$) in the form of a mixed strategy. Since the smallest of Alice's gains in the margins is equal to the largest of Bob's losses in the margins, the value $\frac{ad - bc}{a - b - c + d}$is simultaneously the floor of Alice's gains and ceiling of Bob's losses, fulfilling the definition of a game value. Our solution to any 2×2 game where a-b-c+$d \neq 0$ is then

$$(\langle \frac{d - c}{a - b - c + d}, \frac{a - b}{a - b - c + d} \rangle, \langle \frac{d - b}{a - b - c + d}, \frac{a - c}{a - b - c + d} \rangle, \frac{ad - bc}{a - b - c + d})$$

	Alice\Bob	$\frac{d-b}{a-b-c+d}$ I	$\frac{a-c}{a-b-c+d}$ II	Bob's losses
$\frac{d-c}{a-b-c+d}$	I	a	b	$\frac{ad-bc}{a-b-c+d}$
$\frac{a-b}{a-b-c+d}$	II	c	d	$\frac{ad-bc}{a-b-c+d}$
	Alice's gains	$\frac{ad-bc}{a-b-c+d}$	$\frac{ad-bc}{a-b-c+d}$	$\frac{ad-bc}{a-b-c+d}$

TABLE 5. The updated payoff matrix.

2.3. **Graphical Technique to Solve $2\times n$ and $n\times 2$ Games.** Games with 2×2 payoff matrices are an important foundation of game theory as larger games can sometimes be reduced down to 2×2 games. Consider the payoff matrix in Table 6. In this case, Bob has four pure strategies to choose from. If we can determine that the optimal strategy for Bob is to mix two of his available four strategies, then we can reduce the 2×4 game into a 2×2 game and use the formulas derived in the previous subsection.

		q_1	q_2	q_3	q_4	
Alice\Bob		I	II	III	IV	Bob's losses
p	I	2	-1	3	1	$2q_1 - q_2 + 3q_3 + q_4$
$1-p$	II	-1	4	0	5	$-q_1 + 4q_2 + 0q_3 + 5q_4$
Alice's gains		$2p - 1(1-p)$	$-p + 4(1-p)$	$3p + 0(1-p)$	$p + 5(1-p)$	

TABLE 6. A sample 2×4 payoff matrix.

The optimal strategy for Alice remains finding the floor on her gains. We will call this value u. Alice's marginal values (what she can expect to gain against each of Bob's strategies) should satisfy the following:

$$2p - 1(1-p) \geq u \tag{3}$$

$$-p + 4(1-p) \geq u \tag{4}$$

$$3p + 0(1-p) \geq u \tag{5}$$

$$p + 5(1-p) \geq u \tag{6}$$

We can treat the left hand side of each inequality as a function with respect to p, f(p), and then plot each function as p varies from 0 to 1. This graph be seen in Figure 1. The bold line of Figure 1 is the floor of Alice's gains. We can see that the highest point of the bold line is the intersection of q_1 and q_2. This means that Bob playing a mixed strategy between q_1 and q_2 will give Alice the largest floor for her gains as well as satisfy equations 3 through 6. We have learned from previous examples that finding Alice's floor will dually find Bob's ceiling, so mixing strategies q_1 and q_2 is optimal for Bob as well. Thus, we can reduce the original 2×4 payoff matrix to a 2×2 by keeping the columns of the 2×4 payoff matrix that have strategies q_1 and q_2 over it. We can solve the reduced payoff matrix with the formulas from the previous section. The reduced matrix is seen in Table 7, and solving it will yield the optimal strategies and game value seen in Table 8.

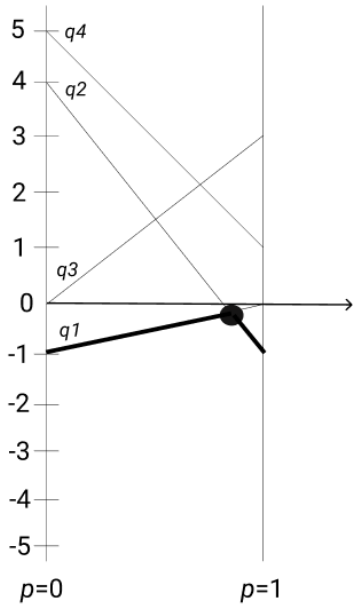

FIGURE 1. The graph of Alice's payoffs against each of Bob's pure strategies.

An interesting note about Table 8 is that we can see how $\frac{7}{8}$ is the floor or minimum of Alice's gains in the bottom margin. Alice does have the potential to gain more than $\frac{7}{8}$ ($\frac{15}{8}$ and $\frac{20}{8}$) if Bob deviates from his optimal strategy and chooses to play his third and fourth strategy. We can say our solution for this 2×4 game is $\left(\langle\frac{5}{8}, \frac{3}{8}\rangle, \langle\frac{5}{8}, \frac{3}{8}, 0, 0\rangle, \frac{7}{8}\right)$. If

		$q_1 = \frac{5}{8}$	$1 - q = \frac{3}{8}$	
	Alice\Bob	I	IV	Bob's Losses
$p = \frac{5}{8}$	I	2	-1	$\frac{7}{8}$
$1 - p = \frac{3}{8}$	II	-1	4	$\frac{7}{8}$
	Alice's Gains	$\frac{7}{8}$	$\frac{7}{8}$	Game Value $= \frac{7}{8}$

TABLE 7. A simplified 2×2 matrix of the original 2×4 game. Notice how the strategies for each player add up to 1.

		$q_1 = \frac{5}{8}$	$q_2 = \frac{3}{8}$	$q_3 = 0$	$q_4 = 0$	
	Alice\Bob	I	II	III	IV	Bob's losses
$p = \frac{5}{8}$	I	2	-1	3	1	$\frac{7}{8}$
$1 - p = \frac{3}{8}$	II	-1	4	0	5	$\frac{7}{8}$
	Alice's gains	$\frac{7}{8}$	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{20}{8}$	Game Value $= \frac{7}{8}$

TABLE 8. The original 2×4 payoff matrix with the optimal strategy filled in.

we were solving for an $n \times 2$ game, then we can use the same graphical method to solve for the solution. The caveat, however, is that we would be looking at the ceiling of Bob's losses, so we would look at the uppermost lines of the graph and find the lowest point of intersection.

3. Implementation of the Simplex Algorithm

In the next three subsections, we learn how the Simplex Algorithm is related to linear programs, the concept of pivoting, and the Column Simplex Method. All are needed to put together the Simplex Algorithm.

3.1. **Linear Programs.**

We currently have techniques to solve 2×2 games, $2 \times n$ games, $n \times 2$ games and any sized game with a Saddle Point Solution. But now we turn our attention to the Simplex Algorithm in

order to solve games with larger game matrices (without a Saddle Point Solution).

The Simplex Algorithm is essentially a technique to solve a linear program. **Linear programming** is used in mathematical optimization when a model's constraints are represented by linear relationships. Let us consider the following linear program:

3.1.1. *Art Sale.* Suppose Alice is planning to sell art. Her two popular items are paintings and scarfs. She will sell paintings for \$5 and scarfs for \$7 and usually makes about \$10 in total tips. Paintings take 2 days to make, while scarfs take 4 days. She only has enough materials to make 5 total items and 12 days to complete them. We will let x_1 represent paintings and x_2 scarfs. The linear program can be represented by the following constraints:

$$x_1 \geq 0 \tag{7}$$

$$x_2 \geq 0 \tag{8}$$

$$x_1 + x_2 \leq 5 \tag{9}$$

$$2x_1 + 4x_2 \leq 12 \tag{10}$$

$$u = 5x_1 + 7x_2 + 10 \tag{11}$$

$$\text{maximize } u$$

Linear programming aims to optimize (finding the maximum or minimum) what is called an **objective function**. We are trying to maximize our profit in this example, and our corresponding objective function is u, which is composed of the variables x_1, x_2, and a constant term. In other words, we are trying to find values for x_1 and x_2 that make the equation u as large as possible but at the same time satisfy the other constraints of the program, which in this case are the constraints 7 through 10.

In order to solve the linear program, we will graph the constraints which are inequalities. For constraints 9 and 10, we can graph them by algebraically turning them into a form resembling $y{=}mx + b$ via treating x_1 like x and x_2 like y. The graph can be seen in Figure 2. We could have simply chose x_1 to be x and x_2 to be y, but the choice of x_1 and x_2 will be helpful for later computations. The shaded region of Figure 2 is called the **feasible region** of the program. The feasible region, its edges and vertices represent solutions to the program, called **feasible solutions** as they satisfy the given constraints. Since our goal is to maximize u, it is a rule of linear programs that the optimal feasible solution can be found at one of the vertices. With some algebraic checks,

we can find that the vertex (4,1) yields the maximum value u=37. Although a graphical approach is fairly straightforward when solving linear programs, the process can get computationally challenging with larger programs that can quickly move away from a 2-D feasible region. This is what motivates the utilization of the Simplex Algorithm as it can handle higher dimensions while still being efficient.

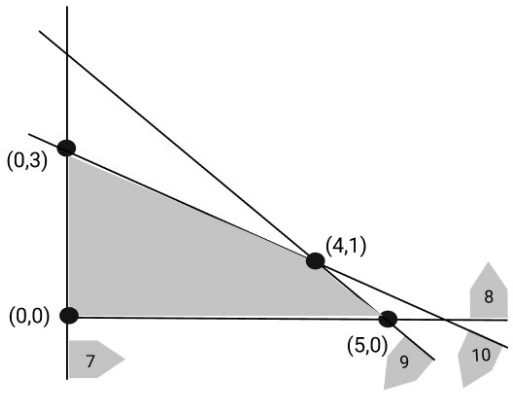

FIGURE 2. Graph of linear program with feasible region shaded.

Let us introduce what are called **slack variables** to inequalities 9 and 10. What slack variables do is allow us to turn an inequality into an equation by adding or subtracting a nonnegative variable which equals out the inequality. In this case, we introduce the slack variables x_3 and x_4.

$$\begin{aligned} x_1 + x_2 + x_3 &= 5 \\ 2x_1 + 4x_2 + x_4 &= 12 \\ x_3 \geq 0, x_4 &\geq 0 \end{aligned}$$

By isolating the slack variables, we can rewrite the linear program in terms of x_1 and x_2:

$$-x_1 - x_2 + 5 = x_3 \tag{12}$$

$$-2x_1 - 4x_2 + 12 = x_4 \tag{13}$$

$$5x_1 + 7x_2 + 10 = u \tag{14}$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

$$\text{maximize } u$$

Here we can set $x_1 = 0$ and $x_2 = 0$ and easily find the values of the slack variables as well as u. By doing so, we see that $x_1 = 0$ and $x_2 = 0$ is a feasible solution to the program which we can write as (x_1,x_2,x_3,x_4) = (0,0,5,12) and $u = 10$. The slack variables don't contribute to the answer for the linear program as we only require the variables we started with, but we will continue to include them and their values. A slack variable that equals 0 indicates that the inequality constraint is at its limit (an edge of the feasible region) [1].

Although the solution when x_1=0 and x_2=0 is feasible, it is not optimal. We can build on the idea of plugging in 0 for two of the variables and solving for the other two, by making it so the values to become 0 are **independent variables**. That is, the values to become 0 are collected on one side of the equation, and the **dependent variable** is on the other side, determined by altering the values of the independent variables. In this example, if we want x_2=0 and x_3=0, then x_2 and x_3 are our independent variables which we want to move to the left hand side in each of the equations, leaving x_1 and x_4 to be dependent variables on the right hand side in their respective equation. Observing equation 12, we see x_2 is already on the left hand side. All that is needed is to algebraically swap x_1 and x_3 and then substitute x_1, which should now be isolated on the right hand side, into the other equations:

$$\begin{aligned} -x_2 - x_3 + 5 &= x_1 \\ -2x_2 + 2x_3 + 2 &= x_4 \\ 2x_2 - 5x_3 + 35 &= u \end{aligned}$$

Now if we let x_2=0 and x_3=0, we get (x_1,x_2,x_3,x_4) = (5,0,0,2) and u = 35 which is another feasible solution. All combinations of setting two out of the four variables (although note that there are much more combinations in larger systems) equal to 0 essentially traverses the vertices of the feasible region where we can then determine the maximum value. This process of solving for a variable and plugging it into the rest of the program, is called an **exchange operation** as we just exchanged the roles of x_1 and x_3. We could continue with exchange operations and eventually determine a feasible solution which finds the max value of u. However, exchange operations can be represented in a more eloquent way which will assist with the Simplex Algorithm.

3.2. **Pivoting.** Let us look at equations 12, 13, and 14, the equations from the program in the last section:

$$\begin{aligned} -x_1 - x_2 + 5 &= x_3 \\ -2x_1 - 4x_2 + 12 &= x_4 \\ 5x_1 + 7x_2 + 10 &= u \end{aligned}$$

Consider Table 9 where we rewrite these equations in a column form. This representation of the program is called a **column program** where the independent variables are along the left side while the dependent (slack) variables are along the bottom.

x_1	-1^*	-2	5
x_2	-1	-4	7
1	5	12	10
	$= x_3$	$= x_4$	$= u$

TABLE 9. The equations written vertically as a matrix with the independent variables in the left column and the dependent variables along the bottom.

If we wanted to repeat the exchange operation as we did in the previous subsection where we solved to have x_2 and x_3 as independent variables on the left hand side, we can utilize Table 9 to algorithmically do the computations, with a technique called **pivoting**. We first need to identify the pivot element in the matrix. Since we want to have x_2 and x_3 as independent variables, and x_2 is already an independent variable, we want to swap the variables x_1 and x_3. Thus, our pivot element will be the element in the same row as x_1 and column as x_3, so -1 which is starred in Table 9. The algorithm for pivoting is as follows:

(1) Replace the nonzero pivot with its reciprocal
(2) Divide all other elements in the pivot's row by the pivot element
(3) Divide all other elements in the pivot's column by the pivot element and then change the sign
(4) For any other element, subtract the element by the product of the element in the same column that is in the pivot's row, the element in the same row that is in the pivot's column, and the reciprocal of the pivot element

(5) For the variable in the side margin in the same row as the pivot element, swap it with the variable in the bottom margin in the same column as the pivot element

x_1	a^*	b	c
x_2	d	e	f
1	g	h	i
	$= x_3$	$= x_4$	$= u$

TABLE 10. A generalized version of Table 9 with the pivot element starred.

x_3	$\frac{1}{a}$	$\frac{1}{b}$	$\frac{1}{c}$
x_2	$\frac{-d}{a}$	e - $\frac{bd}{a}$	f - $\frac{cd}{a}$
1	$\frac{-g}{a}$	h - $\frac{bg}{a}$	i - $\frac{cg}{a}$
	$= x_1$	$= x_4$	$= u$

TABLE 11. The new matrix after the pivot algorithm is applied.

Tables 10 and 11 illustrate pivoting on a generalized matrix. A subtlety of the algorithm is that each step is performed on the original matrix. As an example, you perform Step 1 on Table 10, then Step 2 on Table 10, etc. You do not perform Step 1 on Table 10 and then perform Step 2 on the result of Step 1. Table 12 shows the result after we pivot on the starred element in Table 9.

x_3	-1	2	-5
x_2	-1	-2	2
1	5	2	35
	$= x_1$	$= x_4$	$= u$

TABLE 12. The resulting column program after pivoting. Notice how x_1 and x_3 are swapped from Table 9.

We will call the highlighted row the **distinguished row** and the highlighted column the **distinguished column**. We use the term distinguished to recognize that we do not want to pivot on these elements for computational reasons of the Simplex Algorithm. We define a column program as **column feasible** if all entries in the distinguished

row are nonnegative. Thus, Table 12 is column feasible. As the term feasible suggests, when the matrix is column feasible, it is expressing a feasible solution. In Table 12, $(x_1,x_2,x_3,x_4) = (5,0,0,2)$ and $u = 35$ which is what we saw in the previous subsection.

3.3. **Column Simplex Method and Duality.** In subsection 3.2, we determined that Table 12 exhibited a column feasible solution. However, we know from subsection 3.1 that the maximum value is $u = 37$. Using the **Column Simplex Method**, we can start with the column feasible solution from Table 12 and selectively pivot in order to obtain the optimal feasible solution. The Column Simplex Method is as follows:

(1) Start with a column feasible schema. That is, a column program with all entries in the distinguished row that are nonnegative.
(2) If all entries in the distinguished column are nonpositive (**row feasible**), then the column program exhibits the optimal feasible solution. If this is not true, move on to Step 3.
(3) Without loss of generality, choose a positive element in the distinguished column. We will denote this element with the dummy variable c.
(4) Find all negative elements that are in the same row as c (if there are no negative elements in the same row as c, then the program is unbounded meaning there is no optimal solution). For each of these elements, compute ratios by dividing each negative element from the element in the distinguished row that shares the same column.
(5) Select the element that yields the largest ratio (the ratio closest to 0 on a number line). We will denote this element p.
(6) Pivot using the p as the pivot element. Then go back to Step 2.

As an example, we will use the Column Simplex Method on Table 12:

- We are starting with the column feasible program from Table 12
- The element 2 in the distinguished column is the only nonnegative element, so that is our choice for c
- Both elements in the same row as 2 are negative. The ratio for the first column is $\frac{5}{-1}$. The ratio for the second column is $\frac{2}{-2}$
- $\frac{5}{-1} < \frac{2}{-2}$ so our choice for p is the element -2

- Our pivot element $p = -2$ is in the second row of the second column, and Table 13 shows the result of the pivot

x_3	-2	1	-3
x_4	$\frac{1}{2}$	$\frac{-1}{2}$	-1
1	4	1	37
	$= x_1$	$= x_2$	$= u$

TABLE 13. The column program after pivoting from Table 12 on pivot element -2.

By Step 2 of the Column Simplex Method, we can see that elements in the distinguished column are nonpositive so we have reached the optimal feasible solution which is exactly what we saw in subsection 3.1 where $\langle x_1, x_2 \rangle = \langle 4, 1 \rangle$ on the feasible region in Figure 2 and $u = 37$. Notice that in subsection 3.2, we essentially performed our first iteration of the Column Simplex Method with the element in the top left corner acting as the pivot element.

Recall that what we just solved for was the linear program from subsection 3.1 where we were to maximize the objective function. This translates to how Alice wants to maximize the floor of her gains so we set up a linear program for her and maximized her floor. In a similar manner, we can set up the linear program to minimize the objective function, translating to how Bob wants to minimize the ceiling of his losses. In the next subsection, we will see how we can set up linear programs for Alice and Bob, but the two programs can be represented by the same matrix. By solving this shared matrix, we are simultaneously solving both programs, the essence of dual linear programming or **duality**.

3.4. **Simplex Algorithm.** The Simplex Algorithm puts together tools from the previous subsections which allows us to solve any sized payoff matrix. Let us assemble them with an example:

3.4.1. *Penalty Kicks.* Suppose Alice and Bob are doing penalty kicks in soccer. Alice will kick 5 times while Bob tries to defend them. Alice can either choose to kick to the right side of the goal, left side of the goal, or kick down the middle. Similarly, Bob can defend to the right, left, or middle part the goal. Assume that right, left, and middle part

of the goal is same for the kicker and goalie. Intuitively, when Alice kicks away from where Bob defends, she will receive a favorable chance of scoring. But Bob will receive a favorable chance of blocking the kick if he correctly defends where Alice kicks. We can see these payoffs in Table 14.

		y_4	y_5	y_6
Alice\Bob		R	L	M
x_1	R	4	-3	2
x_2	L	-4	5	0
x_3	M	3	2	-2

TABLE 14. Payoff matrix for Penalty Kicks where Alice is the row player and Bob is the column player.

We strategically denote Bob's pure strategies as $\langle y_4, y_5, y_6 \rangle$ to setup the Simplex Algorithm. Now we can set up our linear program for Alice and Bob:

Alice:

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 4x_1 - 4x_2 + 3x_3 &\geq u \\ -3x_1 + 5x_2 + 2x_3 &\geq u \\ 2x_1 + 0x_2 - 2x_3 &\geq u \\ x_1 \geq 0, x_2 \geq 0, x_3 &\geq 0 \\ \text{maximize } u & \end{aligned}$$

Bob:

$$\begin{aligned} y_4 + y_5 + y_6 &= 1 \\ 4y_4 - 3y_5 + 2y_6 &\leq v \\ -4y_4 + 5y_5 + 0y_6 &\leq v \\ 3y_4 + 2y_5 - 2y_6 &\leq v \\ y_4 \geq 0, y_5 \geq 0, y_6 &\geq 0 \\ \text{minimize } v & \end{aligned}$$

Keep in mind that for the left side of the inequalities in Bob's programs, you take the dot product of his strategies $\langle y_4, y_5, y_6 \rangle$ and each row of the payoff matrix. And because we are trying to minimize Bob's ceiling, the inequalities are less than or equal to v, which is what we will denote as the ceiling of Bob's losses. We can introduce slack variables to the inequalities of each program:

Alice	Bob
$4x_1 - 4x_2 + 3x_3 - x_4 = u$	$4y_4 - 3y_5 + 2y_6 + y_1 = v$
$-3x_1 + 5x_2 + 2x_3 - x_5 = u$	$-4y_4 + 5y_5 + 0y_6 + y_2 = v$
$2x_1 + 0x_2 - 2x_3 - x_6 = u$	$3y_4 + 2y_5 - 2y_6 + y_3 = v$
$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$	$y_4 \geq 0, y_5 \geq 0, y_6 \geq 0$
$x_4 \geq 0, x_5 \geq 0, x_6 \geq 0$	$y_1 \geq 0, y_2 \geq 0, y_3 \geq 0$

Notice that for Alice's program, we subtract the slack variables since we need to reduce the value of the left hand side of the inequality to make equations. We will rewrite the programs with the slack variables as dependent variables:

Alice	Bob
$x_1 + x_2 + x_3 = 1$	$y_4 + y_5 + y_6 = 1$
$4x_1 - 4x_2 + 3x_3 - u = x_4$	$4y_4 - 3y_5 + 2y_6 - v = -y_1$
$-3x_1 + 5x_2 + 2x_3 - u = x_5$	$-4y_4 + 5y_5 + 0y_6 - v = -y_2$
$2x_1 + 0x_2 - 2x_3 - u = x_6$	$3y_4 + 2y_5 - 2y_6 - v = -y_3$
$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$	$y_4 \geq 0, y_5 \geq 0, y_6 \geq 0$
$x_4 \geq 0, x_5 \geq 0, x_6 \geq 0$	$y_1 \geq 0, y_2 \geq 0, y_3 \geq 0$
maximize u	minimize v

At this point, we could set up a column program for Alice and a **row program** for Bob. The schematic of a row program with two inequality constraints is shown in Table 15. In duality to column programs, row programs can be row feasible where all elements in the distinguished column are nonpositive (except for possibly the bottom element). There is also a **Row Simplex Method** to use if the row program is row feasible. However, the magic of duality allows us to only have to use either the Column Simplex Method or Row Simplex Method, as they both lead us to the same result. We will choose to use the Column Simplex Method.

We will now represent Alice's column program (and simultaneously Bob's row program) in a schematic form introduced by mathematician A.W. Tucker [4]. It is shown in Table 16. Notice that the highlighted partition is just the original game matrix! Thus, when the Simplex Algorithm is given a game matrix, the game matrix goes in the middle and the appropriate number of 1's and -1's are added to the margins along with the appropriate number of variables. Towards the end of

y_3	y_4	1	
a	d	g	$= -y_1$
b	e	h	$= -y_2$
c	f	i	$= v$

TABLE 15. Schematic for a row program. Later we will swap independent and dependent variables. Variables swapped to the top turn positive while variables swapped to the right turn negative.

subsection 3.3, it is discussed how the programs created by the constraints of each player can be represented by the same matrix. That matrix is exactly Table 16! The coefficients for Alice's program are written as columns with the variables representing Alice's strategy on the left margin and the slack variables along the bottom margin. The coefficients for Bob's program are written as rows with the variables representing Bob's strategy along the top margin and the slack variables on the right margin.

	$-v$	y_4	y_5	y_6	
u	0	-1	-1	-1	$= -1$
x_1	1	4	-3	2	$= -y_1$
x_2	1	-4	5	0	$= -y_2$
x_3	1	3	2	-2	$= -y_3$
	$= 1$	$= x_4$	$= x_5$	$= x_6$	

TABLE 16. A.W. Tucker's schematic form to set up the Simplex Algorithm. Highlighted in cyan is the original payoff matrix.

The Simplex Algorithm now wants us to pivot on the bottom left corner of the schematic form (Table 17) and then pivot on the top right corner (Table 18). After the two pivots, we can redraw the dashed lines to get the highlighted distinguished row and column in Table 19. It is at this point in the Simplex Algorithm that the program

will either be column feasible or row feasible. Since the elements in the distinguished row are all nonnegative, the highlighted partition is column feasible, and we can use the Column Simplex Method until all the elements in the distinguished column are nonpositive. In other words, we have found the optimal solution when the program is column and row feasible.

	$-v$	y_4	y_5	y_6	
u	0	-1	-1	-1	$=-1$
x_1	1	4	-3	2	$=-y_1$
x_2	1	-4	5	0	$=-y_2$
x_3	1*	3	2	-2	$=-y_3$
	$=1$	$=x_4$	$=x_5$	$=x_6$	

TABLE 17. Begin by pivoting on the bottom left corner.

	y_3	y_4	y_5	y_6	
u	0	-1	-1	-1*	$=-1$
x_1	-1	1	-5	4	$=-y_1$
x_2	-1	-7	3	2	$=-y_2$
1	1	3	2	-2	$=v$
	$=x_3$	$=x_4$	$=x_5$	$=x_6$	

TABLE 18. The result of the first pivot. Now we must pivot on the top right corner.

It is not by coincidence that this example yielded a column feasible program (seen in Table 19). It is a result of the fact that element in the bottom right corner of the original payoff matrix was the the smallest element in its row. If we wanted the program to be row feasible after the two pivots (and hence use the Row Simplex Method), then the element in the bottom right corner of the original matrix needs to be the largest element in its column.

	y_3	y_4	y_5	1	
x_6	0	1	1	-1	$= -y_6$
x_1	-1	-3	-9	4	$= -y_1$
x_2	-1	-9	1	2	$= -y_2$
1	1	5	4	-2	$= v$
	$= x_3$	$= x_4$	$= x_5$	$= u$	

TABLE 19. The result of the second pivot.

Let us continue with the example:

- We need to find the the pivot element of the program in Table 19. The Column Simplex Method says to pick a nonnegative element, so we can choose $c = 4$ or $c = 2$. We will end up at the same answer either way, so we will choose $c = 4$
- All elements in the same row as $c = 4$ are negative, so all columns are being considered in the ratio comparison. $\frac{5}{-3} < \frac{1}{-1} < \frac{4}{-9}$. So we will pivot on the third column, and our pivot element will be $p = -9$. The result of pivoting is shown in Table 20
- Looking at Table 20, there is still one more value in the distinguished column that has a nonnegative value, so we will perform another iteration Column Simplex Method with $c = \frac{22}{9}$. There are only two negative elements in the same row as c. Comparing the ratios, we have $\frac{-5}{10} < \frac{-11}{28}$. So $p = \frac{-28}{3}$. The result of pivoting is shown in Table 21

Looking at Table 21, all values in the distinguished row are nonnegative, and all values in the distinguished column are nonpositive. Since the program is now column and row feasible, we have reached the optimal solution. All variables along the left and top are independent variables, so recall from subsection 3.1 that they equal 0.

	y_3	y_4	y_1	1	
x_6	$\frac{-1}{9}$	$\frac{2}{3}$	$\frac{1}{9}$	$\frac{-5}{9}$	$= -y_6$
x_5	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{-1}{9}$	$\frac{-4}{9}$	$= -y_5$
x_2	$\frac{-10}{9}$	$\frac{-28}{3}$	$\frac{1}{9}$	$\frac{22}{9}$	$= -y_2$
1	$\frac{5}{9}$	$\frac{11}{3}$	$\frac{4}{9}$	$\frac{-2}{9}$	$= v$
	$= x_3$	$= x_4$	$= x_1$	$= u$	

TABLE 20. The result after pivoting on $p = -9$.

Here are the optimal strategies for Alice and Bob:

$$\text{Alice: } \langle x_1, x_2, x_3, x_4, x_5, x_6 \rangle = \langle \frac{41}{84}, \frac{11}{28}, \frac{5}{42}, 0, 0, 0 \rangle \text{ and } u = \frac{31}{42}$$
$$\text{Bob: } \langle y_1, y_2, y_3, y_4, y_5, y_6 \rangle = \langle 0, 0, 0, \frac{11}{42}, \frac{5}{14}, \frac{8}{21} \rangle \text{ and } u = \frac{31}{42}$$

The values are rewritten in terms of the original matrix in Table 22. In the context of the 5 penalty kicks, Alice should kick to the right $5(0.49) \approx 2$ times, to the left $5(0.39) \approx 2$ times, and the middle $5(0.12) \approx 1$ time. It does not matter what order Alice kicks in, as long as she follows the number of kicks per direction, she is playing optimally. Bob should defend to the right $5(0.26) \approx 1$ time, to the left $5(0.36) \approx 2$ times, and the middle $5(0.38) \approx 2$ times. In a similar fashion, it does not matter what order Bob defends in, as long as he follows the number of dives per direction, he is playing optimally. The game value is about 0.74, so 0.74 is the floor of Alice's gains and ceiling of Bob's losses. Along the bottom edge, we can see the game value or a value

	y_3	y_2	y_1	1	
x_6	$\frac{-4}{21}$	$\frac{1}{14}$	$\frac{5}{42}$	$\frac{-8}{21}$	$= -y_6$
x_5	$\frac{1}{14}$	$\frac{1}{28}$	$\frac{-3}{28}$	$\frac{-5}{14}$	$= -y_5$
x_4	$\frac{5}{42}$	$\frac{-3}{28}$	$\frac{-1}{84}$	$\frac{-11}{42}$	$= -y_4$
1	$\frac{5}{42}$	$\frac{11}{28}$	$\frac{41}{84}$	$\frac{31}{42}$	$= v$
	$= x_3$	$= x_2$	$= x_1$	$= u$	

TABLE 21. The result after pivoting on $p = \frac{-28}{3}$.

Alice\Bob	$y_4 = 0.26$	$y_5 = 0.36$	$y_6 = 0.38$	Bob's Losses
$x_1 = 0.49$	4	-3	2	0.74
$x_2 = 0.39$	-4	5	0	0.74
$x_3 = 0.12$	3	2	-2	0.74
Alice's Gains	0.74	0.74	0.74	$V = 0.74$

TABLE 22. The Penalty Kicks payoff matrix with the probabilities that each player should play with and the marginal values.

higher than the game value, because the game value is the best Alice can guarantee for herself, but she has the potential to gain more as we saw in subsection 2.3. Along the right edge, we can see the game value or a value lower than the game value, because the game value is the guaranteed worst that Bob can lose, but he has the potential to lose

less if he plays optimally. In this case, however, 0.74 is lowest Alice can expect to gain and the highest Bob can expect to lose.

Something to note about the optimal strategies for each player is that it does not matter if we played 5 games (like we did for Penalty Kicks), 10 games, or 100 games. The optimal strategies represent probabilities (proportions) of the time a player should play each strategy. If we were to do 100 penalty kicks and each kick is represented by the payoff matrix in Table 22, then Alice can determine how many times to kick right by taking $100(0.49) = 49$ kicks, kick left by taking $100(0.39) = 39$ kicks, and kick to the middle by taking $100(0.12) = 12$ kicks. The same can be found for Bob.

To summarize the **Simplex Algorithm**:

(1) Given a game matrix, label it with variables representing the strategies for the row and column player. Check the bottom row to see if the element in the right corner is a row minimum. If it is, move to the next step. If it is not, find the column that has the the row minimum for the bottom row and swap that column with the last column. Be sure to swap the corresponding variables if you do so. Then move to the next step.
(2) Put the game matrix into A.W. Tucker's schematic form.
(3) Pivot on the bottom left corner.
(4) Pivot on the top right corner.
(5) As a consequence of the bottom right corner being a row minimum, the program will be column feasible. If it also row feasible, move to the next step. If not, iteratively use the Column Simplex Method until the program is column and row feasible. Then move to the next step.
(6) Interpret the results by rewriting the probabilities in the original game matrix.

3.5. **Simplex Algorithm Implementation in Mathematica.** With assistance from David Roberts, I implemented the Simplex Algorithm in the Mathematica software. All that is needed is to input a matrix into the function `simplexAlgorithm[matrix_]` and the solution of the game is returned.

As an example, consider the game matrix from subsection 3.4 and the result when put into the function `simplexAlgorithm[matrix_]`:
`{GameValue=31/42, p_1=41/84, p_2=41/84, p_3=5/42, q_1=11/42, q_2=5/14, q_3=8/21}`. This is exactly the result we previously calculated!

The function `simplexAlgorithm[matrix_]` has various helper functions to incorporate all the sub algorithms and computations covered in this paper:

(1) The input matrix is passed as input to a function called `alterMatrix[matrix_]`. It creates the variables representing the strategies for each player as well as slack variables. Then it checks to see if the bottom right corner of the game matrix is a row minimum. If it already is, then it outputs the matrix and all the variables as a pair. If not, then it swaps the last column of the matrix with the column holding the smallest element in the bottom row. Then it outputs the aforementioned pair
(2) The pair from the output of `alterMatrix[matrix_]` is passed into the function `initialize[pair_]`. This function takes the matrix from the pair and converts it into A.W. Tucker's schematic form. Then it does a pivot on the bottom left corner, and the top right corner which sets us up to use the Column Simplex Method. The variables are also swapped accordingly, then the new matrix and shuffled variables are output as a pair
(3) The pair from the output of `initialize[pair_]` is passed into the function `columnSimplex[pair_]`. This function takes the matrix from the pair and sees if all elements in the distinguished column are nonpositive (row feasible). If it is not, it recursively performs the Column Simplex Method until the matrix is row feasible. During each iteration of the Column Simplex Method, the variables are swapped accordingly as pivoting is required. When the matrix finally reaches row feasibility, it and the further shuffled variables are output as a pair
(4) The rest of the function `simplexAlgorithm[pair_]` then uses the output pair of matrix and variables to return to the user the game value and optimal strategies for each player

Before implementing the Simplex Algorithm, I had previously implemented the Shapley-Snow Algorithm in Mathematica. It is another technique used to solve two player zero sum matrix games, but it is far less efficient. It is a known fact of game theory that optimal strategies involve the same number of strategies for both players. As an example, if we consider a 5×7 matrix game and the optimal strategy for Alice uses three of her strategies, then Bob's optimal strategy also uses three of his strategies. Based on this, we can assume that the optimal strategy for each player can have up to five strategies. My Shapely-Snow implementation looks at the case when each player uses 5 strategies and sees if the optimal strategy exists there. If not, then it moves on

to when each player uses 4 strategies, then 3 strategies, etc. until it finds the optimal strategy. This idea of a blind search is vastly inefficient, and it motivates us to utilize the algorithmic approach of the Simplex Algorithm.

4. Conclusion

In this paper, we have seen how we can solve two person zero sum games by hand and with the Simplex Algorithm. The Simplex Algorithm is an especially nifty technique compared to others such as the Shapley-Snow Algorithm (or even just solving by hand) because of its efficiency. Its utilization continues to be widely popular in Mathematics, Economics, and decision-making.

Despite its usefulness, there are still some drawbacks and limitations to be explored. The most pressing issue in my research of the Simplex Algorithms are ties in the game matrix, when the same row or column shared the same value. When there were ties, sometimes I ran into computational errors in Mathematica, so that is an area of concern to be addressed. Another point of interest are the limitations of Mathematica's computational ability. After enlarging Mathematica's recursion limit, I found that the implementation was able to solve a 100×100 matrix game in about 10 minutes, which is the largest payoff matrix I have tested. Undoubtedly, there is more than one way to implement the Simplex Algorithm, with some being more efficient than others. However, the notion of constrained size and computational power is still seemingly a barrier to be overcome.

5. Acknowledgements

I would like to thank David Roberts for his guidance and inspiration throughout the research process along with Chris Atkinson and all the math faculty at UMM for taking the time to help me with this paper. My research was primarily from [4] where the authors explained how the Simplex Algorithm and linear programs are related.

References

[1] APMonitor. "Slack Variable Tutorial." APMonitor, June. 2013, http://apmonitor.com/wiki/index.php/Main/SlackVariables

[2] Dina Zuko. "The Unconscious, Emotions, and Our Decision-Making Process." UX Collective, UX Collective, 25 Feb. 2020, https://uxdesign.cc/the-unconscious-emotions-and-our-decision-making-process-183002021a29.

[3] "Game Theory." Wikipedia, Wikimedia Foundation, 13 Mar. 2022, https://en.wikipedia.org/wiki/Game_theory.

[4] Singleton, Robert E., and William F. Tyndall. Games and Programs: Mathematics for Modeling. Freeman, 1975.