Python Package for Finding Nash Equilibrium

1st Son Pham
dept. Aerospace Engineering Sciences
University of Colorado, Boulder
Boulder, Colorado
son.pham-2@colorado.edu

***Abstract*—This project develops a comprehensive Python package, *Nashypy*, designed to analyze Nash equilibria in strategic games, facilitating the study of game theory's fundamental concepts. The core functionality of the package includes the ability to read and process game data from both external files and pre-populated examples, compute best responses, and identify both pure and mixed Nash equilibria. Key features of the package involve the application of numerical methods and symbolic computation using libraries such as NumPy and SymPy, which allow for the handling of 2x2 bimatrix games. The package also includes visualization capabilities to graphically display the best response polyhedra and strategy dynamics, enhancing the user's ability to analyze and interpret the strategic structure of games. The utility of this tool, which is currently in development, extends to educational environments, where it can be used to demonstrate interactive game theory analyses, and in research, where complex game theoretical models can be explored and visualized.**

***Index Terms*—game theory, lrsnash, lexicographic reverse search**

I. Introduction

This project is dedicated to the development of a robust Python package that applies algorithms for the enumeration of all Nash equilibria in two-player (bimatrix) games. Inspired by the methodologies outlined in the paper "Enumeration of Nash Equilibria for two-player games," the focus will be on the implementation of the lrsNash algorithm. The primary challenge involves accurately computing equilibria for both non-degenerate and degenerate games. The initial phase will involve developing a basic version of the lrsNash algorithm capable of resolving small-scale bimatrix games, such as 2x2 games. This foundational implementation will serve to validate the core algorithmic structure and establish the precision required for computing Nash equilibria. The implementation strategy will incorporate the lrsNash algorithm specifically for bimatrix games, focusing on the integration of best response conditions and the enumeration of equilibria through labeled polytopes.

Key components of this approach include incorporation of fundamental bimatrix game concepts and computational methods for determining best responses based on payoff matrices, facilitating the identification of strategic equilibria.Implementation of functions to visualize best response polyhedra using Plotly and Matplotlib, allowing users to graphically explore the strategic options and equilibria directly from the game's payoff matrices. This package will be able to handle both non-degenerate and degenerate games, employing specific algorithmic adjustments to compute Nash equilibria in varied game scenarios.

II. Background and Related Work

The paper 'Enumeration of Nash Equilibria for Two-Player Games' by Avis et al [1] presents two algorithms, lrsNash and a modified EEE, for finding all Nash equilibria in two-player games in strategic form. It introduces these algorithms within a unified framework based on best-response polyhedra, and discusses potential extensions to other forms. Additionally, the paper details implementation specifics and compares the computational efficacy of the algorithms through experiments.

The paper "Computing Equilibria in Bimatrix Games by Parallel Vertex Enumeration" by Jonathan Widger and Daniel Grosu [2] explores the design and implementation of a parallel algorithm for computing Nash equilibria in bimatrix games. This approach uses vertex enumeration of best response polyhedrons on a grid computing system to efficiently solve the equilibria problem for two-player general-sum normal form games. The performance of the algorithm is analyzed through extensive experiments, demonstrating its effectiveness in handling large-scale games by leveraging parallel computing architectures.

Bernhard von Stengel's work, "Computing Equilibria for Two-Person Games," [3] presents a comprehensive overview of linear methods to find Nash equilibria in two-person games. The paper delves into algorithms like the Lemke-Howson algorithm and explores their geometric and algebraic computation through best-response polyhedra. It also addresses the challenges posed by degenerate games and various equilibrium refinements.

The paper "Computing Lexicographically Safe Nash Equilibria in Finite Two-Person Games with Tight Game Forms Given by Oracles" by Vladimir Gurvich and Mariya Naumova [4] explores a novel proof for the Nash-solvability of tight two-person games and introduces a polynomial algorithm for computing Nash Equilibria that are lexicographically safe. These equilibria are categorized into two types based on one player's lexsafe strategy and the other's special best response. The research extends to the analysis of game forms represented by oracles, highlighting their practical applications in solving zero-sum and ±1 games efficiently.

The paper "Deep Reinforcement Learning for Nash Equilibrium of Differential Games" by Zhenyu Li and Yazhong Luo [5] develops two deep reinforcement learning algorithms

to solve Nash equilibria in differential games. These games involve dynamic conflicts among multiple players where traditional methods struggle due to complex dynamics. This research connects directly with the project's aim of leveraging advanced computational techniques to analyze Nash equilibria in strategic games, offering potential methods for addressing complex dynamic game scenarios with better computational efficiency.

A. Problem Formulation

This project aims to develop a robust package solution designed to compute Nash equilibria in bimatrix games, with potential extensions to more complex game configurations. The problem of finding Nash equilibria, especially in games that can be both non-degenerate and degenerate, poses significant computational challenges due to the complexity of the equilibrium landscape and the mathematical algorithms involved. The functions are designed to systematically find all Nash equilibria in bimatrix games, leveraging a reverse search methodology that integrates several processes discussed below.

Initialization and Data Handling: The package begins with the **read_game** function, which is essential for loading game data either from local files or packaged examples. This function sets the stage for all subsequent computations by initializing **NashGame** objects with defined payoffs for two players, facilitating a structured approach to analyzing bimatrix games. The file format for a stag hunt game typically involves two matrices representing the payoffs for Player 1 and Player 2, respectively. Here is an example of a stag hunt game involving two payoff matrices for Player 1 and Player 2:

```
4 1
3 2

4 3
1 2
```

- **First Matrix (Player 1's payoffs)**: This matrix represents the payoffs for Player 1 when choosing between two strategies (Stag or Hare). The elements of the matrix are:

$$\begin{bmatrix} 4 & 1 \\ 3 & 2 \end{bmatrix}$$

  where the payoff of 4 (top left) is received when both players choose Stag, and 1 (top right) when Player 1 chooses Stag and Player 2 chooses Hare, etc.
- **Second Matrix (Player 2's payoffs)**: This matrix is analogous to the first but for Player 2. The elements are:

$$\begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$$

  where the payoff of 4 (top left) is received when both players choose Stag.

Function Interactions and Workflow

1) **find_nash_equilibria**: Initiates the process by defining starting vertices based on the pure strategies available for both players. Each vertex represents a potential Nash equilibrium candidate defined in the subsection below.
2) **reverse_search**: For each starting vertex, a deep search is conducted to explore all strategic combinations stemming from the initial point. This function uses a stack to manage the search process, checking each strategy pair for equilibrium status.
3) **is_nash_equilibrium**: This function evaluates whether a given strategy pair forms a Nash equilibrium by comparing the expected payoffs to the best response payoffs. Only strategy pairs that satisfy the Nash conditions are considered further.
4) **generate_and_sort_strategies**: As new strategy pairs are identified during the reverse search, they are directly generated and sorted. This ensures that all potential strategies are systematically explored and that the search space is efficiently managed.
5) **solve_mixed_nash**: This function, independent from the main functions of the tool, handles the computation of Nash equilibria for both 2x2 and 3x3 game matrices by determining mixed and pure strategies. For 2x2 games, it calculates potential mixed strategies by examining differences in payoffs and checks for equilibria. For 3x3 matrices, it expands to solve systems of equations representing equilibrium conditions using symbolic computation.
6) **plot_best_response_polyhedra**: This function visualizes the best response polyhedra for each player. It graphs players' payoffs as functions of their strategy mixes and highlights the best responses. This visualization aids in understanding strategic interactions and the consequences of different strategy choices.
7) **plot_best_response_polytope**: Under development, this function provides visual representations of strategic possibilities and constraints via polytopes. It aims to provide the enumerations needed for the application of lrsNash algorithm, which relies on lexicographic reverse searching through vertices of strategy polytopes to enumerate all Nash equilibria in a game.

Best Response Conditions

Best response conditions for Nash equilibrium are defined as follows:

Let x and y be mixed strategies of player 1 and 2, respectively. Then x is a best response to y if and only if for all $i \in M$ [1],

$$x_i > 0 \implies (Ay)_i = u = \max\{(Ay)_k \mid k \in M\}, \quad (1)$$

and y is a best response to x if and only if for all $j \in N$,

$$y_j > 0 \implies (B^T x)_j = v = \max\{(B^T x)_k \mid k \in N\}. \quad (2)$$

Lexicographical Sorting

The reverse search algorithm implemented in this project differs from David Avis's lrsNash algorithm, which utilizes lexicographic reverse search. Our method conducts a direct,

depth-first exploration of strategies, sorting them based on matrix indices primarily for organizational efficiency. In contrast, lrsNash employs lexicographical ordering and pivotal moves similar to simplex methods in linear programming, specifically designed to traverse polytope vertices systematically. This approach allows lrsNash to focus on minimal lexicographical strategies, whereas this project's algorithm aims for comprehensive exploration without prioritizing lexicographical minimality. Currently, the project supports polyhedra functionality, and the development of polytope functions is underway, expected to be implemented in future updates as it requires additional development hours.

III. Solution Approach

The approach to finding Nash equilibria in the package involves several steps that integrate numerical and analytical methods to explore nash equilibria:

1) **Reading and Processing Game Data:** The system initiates by reading game payoff matrices from specified files or examples. These matrices are parsed to extract the strategic payoff information for two players, which are then used to form the basis of Nash equilibrium calculations.
2) **Reverse Search Method:** Employing a reverse search algorithm, the system explores all possible strategy combinations. This method checks each strategy against the Nash equilibrium conditions, where strategies are validated based on their payoff outcomes compared to the best possible responses.As part of the reverse search, the number of vertices (strategy combinations) visited during the search is counted.
3) **Mixed Strategy Analysis:** For games that contain mixed strategies, the solve_mixed_nash function computes equilibria involving probabilistic strategy mixes. This function solves systems of equations derived from the game's payoff matrices, determining equilibrium strategies that may include mixed strategy responses.
4) **Visualization of Strategies:** The package incorporates functions like plot_best_response_polyhedra and 'plot_best_response_polytope' to visually represent the strategic interactions and their respective payoffs. These computations are meant to help build the lexicographic arrays like in lrsNash, but also helps visualize the influence of different strategies on the game's outcome.

IV. Results

Results are presented for a few example games: 2x2 chicken game, 2x2 zero sum game, 2x2 degenerate game, and a 3x2 game from [1] and [2].

Nash Equilibria for a 2x2 Chicken Game

Table I and Figures 1 and 2 present the results of analyzing Nash equilibria in a 2x2 game of chicken. The table summarizes the Nash equilibria found, including pure strategies where each player chooses a dominant strategy leading to payoffs of (4,0) and (0,4), reflecting the typical 'dare' or 'chicken out' decisions in game theory. Additionally, it presents a mixed strategy equilibrium where both players randomize their strategies with probabilities p=0.75 and q=0.75, showing a more balanced approach to risk and payoff. Figure 1 shows the best response polyhedra for Player 1, plotting possible payoffs against different strategy mixes of Player 2. It illustrates the changes in Player 1's best response as Player 2 varies their strategy. Highlighted in green is the best response polyhedron for Player 1.

Strategy 1	**Strategy 2**	**Payoffs**
(1.0, 0.0)	(0.0, 1.0)	(4.0, 0.0)
(0.0, 1.0)	(1.0, 0.0)	(0.0, 4.0)
Total vertices visited		8
Mixed Strategy		{'p': 0.75, 'q': 0.75}

TABLE I
NASH EQUILIBRIA FOR 2X2 CHICKEN GAME

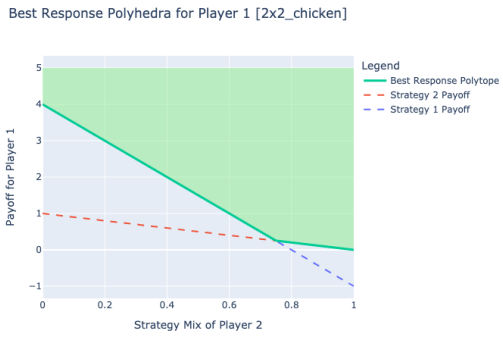

Fig. 1. Polyhedra for Player 1 in the 2x2 Chicken game

Figure 2 illustrates the best response probabilities for Player 1 in a strategic game setting. The x-axis represents the probability of the opponent (Player 2) playing Strategy 1, while the y-axis shows the probability of Player 1's best response. The shaded regions indicate the preferences for Player 1's strategies: the blue area (Strategy 1) is preferred when the opponent predominantly plays Strategy 2, and the orange area (Strategy 2) is preferred when the opponent predominantly plays Strategy 1. The intersection lines highlight points where Player 1 is indifferent between the two strategies, effectively mapping out the best response dynamics based on the opponent's mix of strategies.

Nash Equilibria for a 2x2 Zero Sum Game

The Nash equilibria found are listed as follows:

Mixed Strategy	$\{p : 0.5, q : 0.5\}$
Total vertices visited	8

TABLE II
NASH EQUILIBRIA FOR ZERO SUM GAME

Figure 3 and Table II depict the results of Nash equilibrium analyses for a 2x2 Zero Sum Game. In Figure 3, the Best Response Polyhedra for Player 1 illustrates the payoff outcomes across different strategy mixes of Player 2. The

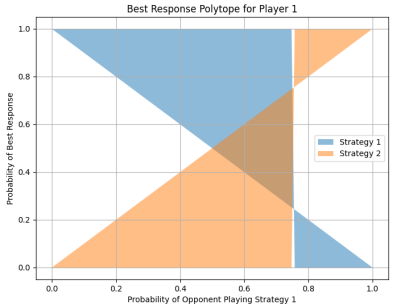

Fig. 2. Probabilities for Player 1 in the 2x2 Chicken game

graph demonstrates that Player 1's optimal strategies change as Player 2 adjusts their strategy mix. In 4, the probabilities for Player 1 show areas where each of Player 1's strategies is a best response to Player 2's strategy. The diagrams indicate that the game is symmetric, meaning the analysis and conclusions apply equally to both players. The mixed strategy Nash equilibrium is represented by both players randomizing their strategies, with each strategy having a probability of 0.5. This symmetric nature and the visualization highlight the strategic balance each player must maintain to achieve equilibrium in this zero-sum context.

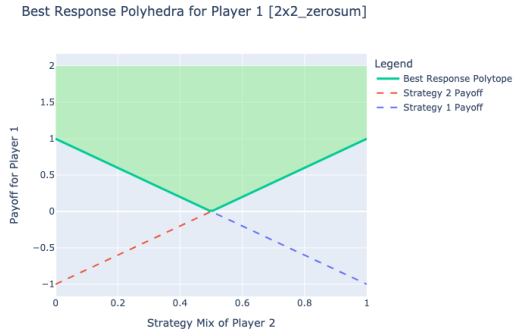

Fig. 3. Polyhedra for Player 1 in the 2x2 Zero Sum Game

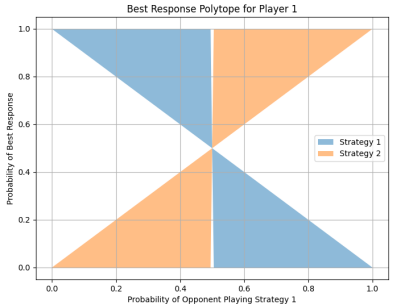

Fig. 4. Probabilities for Player 1 in the 2x2 Zero Sum Game

Nash Equilibria for a 2x2 Degenerate Game

The Nash equilibria found are listed as follows:

Strategy 1	Strategy 2	Payoffs
(1.0, 0.0)	(0.0, 1.0)	(5.0, 4.0)
(0.0, 1.0)	(1.0, 0.0)	(2.0, 5.0)
Total vertices visited		8

TABLE III
NASH EQUILIBRIA FOR D. AVIS GAME

Table III show results for a 2x2 degenerate game given in D. Avis' example [1]. Figure 5 illustrates the best response polyhedra for Player 1 in a degenerate game setting where strategic interactions yields are based on the strategy mix of Player 2. The green shaded area represents the polytope of player 2. The diagonal line represents the change in Player 1's payoff as Player 2 varies their strategy. Figure 6 displays the best response polyhedra for Player 2, reflecting how Player 2's payoffs vary with changes in Player 1's strategy mix. The green region is the polytope for Player 1. Figure 7 and Figure 8 shows the best response polytope given the inequality constraints for player 1 and 2, respectively.

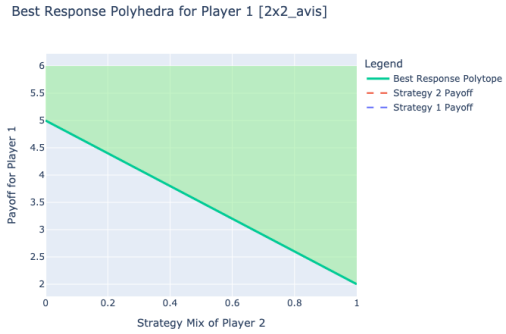

Fig. 5. Polyhedra for Player 1 in the D. Avis Game Example

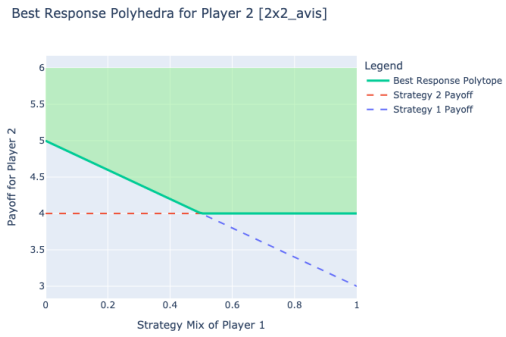

Fig. 6. Polyhedra for Player 2 in the D. Avis Game Example

Nash Equilibria for a 3x2 Game

The Nash equilibria found are listed as follows:

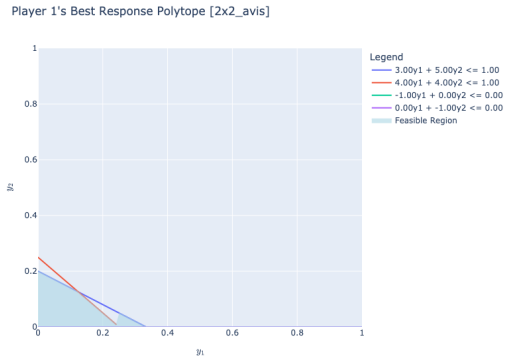

Fig. 7. Polytope for Player 1 in the D. Avis Game Example

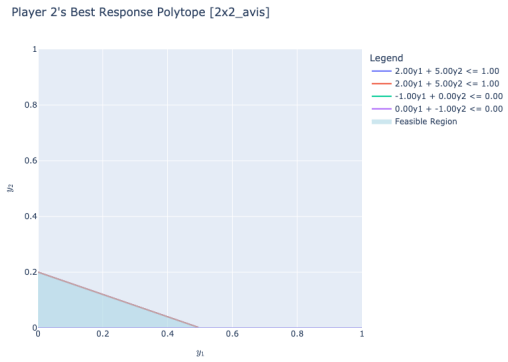

Fig. 8. Polytope for Player 2 in the D. Avis Game Example

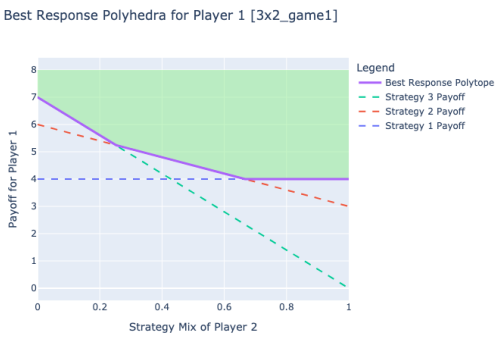

Fig. 9. Polyhedra for Player 1 in a 3x2 Game

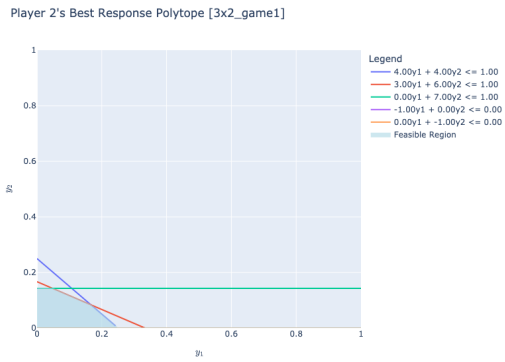

Fig. 10. Polytope for Player 2 in a 3x2 Game

Figure 9 plot displays the polyhedra for Player 1's best responses based on the strategy mix of Player 2 in a 3x2 game scenario where strategy overlaps and payoff efficiencies are evident. The green region is the best response polytope for player 2. The solid line represents the boundary of optimal responses as Player 2's strategy varies. Each dashed line corresponds to the payoff Player 1 receives for committing to one of three strategies regardless of Player 2's actions, showing a dynamic shift in strategy utility across the range.

Figure 10 shows the polytope for player 2 as functions of the inequality constraints for the problem as given in Widger [2]. This polytope illustrates feasible strategy mixes for Player 2 when Player 1 adopts certain strategies in a 3x2 game. The plot uses linear constraints derived from the game's payoff matrix to delineate regions representing viable strategic combinations that Player 2 can adopt. The thin lines define boundaries where the payoff to Player 2 changes.

V. Conclusion

Nashpy is an application of game theory to strategic games through its functionalities designed for analyzing Nash equilibria. By incorporating numerical methods and symbolic computation from external python packages, it has the potential to handle complex bimatrix game scenarios with precision. Visualization features implemented enhance user engagement by allowing visual exploration of strategic interactions through graphical displays of best response polyhedra and polytopes.

The initial stages have focused on setting up a basic framework capable of handling small-scale bimatrix games, laying the groundwork for more complex implementations. Future expansions include fully integrating the lrsNash algorithm for lexicographic reverse searching and developing the modified EEE method to enhance the efficiency of Nash equilibrium computations. Additionally, leveraging parallel processing techniques as discussed by Widger [2] could address the computational demands of larger and more complex strategic setups, improving performance and scalability.

In conclusion, this project not only opens up functionality of game theory in Python but also sets the stage for potential future developments. The next steps involve refining the integration of complex enumerative methods and expanding the package's capabilities to handle a broader range of game theoretical models. This ongoing development aims to open theoretical game analysis into a more accessible and practical tool for strategic decision-making in various fields.

Release

The authors grant permission for this report to be posted publicly. However, please be advised that the tool described herein is currently in development. As such, it may contain

incomplete features and is subject to errors due to ongoing testing and refinement processes. Users should exercise caution and not rely solely on this tool for critical decisions until a stable version has been released.

References

References

[1] D. Avis, G. D. Rosenberg, R. Savani, and B. von Stengel, "Enumeration of nash equilibria for two-player games," Economic Theory, vol. 42, no. 1, pp. 9–37, Mar. 2009. doi:10.1007/s00199-009-0449-x

[2] J. Widger and D. Grosu, "Computing equilibria in Bimatrix games by parallel vertex enumeration," 2009 International Conference on Parallel Processing, Sep. 2009. doi:10.1109/icpp.2009.11

[3] B. Von Stengel, "Chapter 45 computing equilibria for two-person games," Handbook of Game Theory with Economic Applications, pp. 1723–1759, 2002. doi:10.1016/s1574-0005(02)03008-4

[4] V. Gurvich and M. Naumova, "Computing lexicographically safe Nash equilibria in finite two-person games with tight game forms given by oracles," Discrete Applied Mathematics, vol. 340, pp. 53–68, Dec. 2023. doi:10.1016/j.dam.2023.06.030

[5] Z. Li and Y. Luo, "Deep reinforcement learning for nash equilibrium of Differential Games," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–15, 2024. doi:10.1109/tnnls.2024.3351631