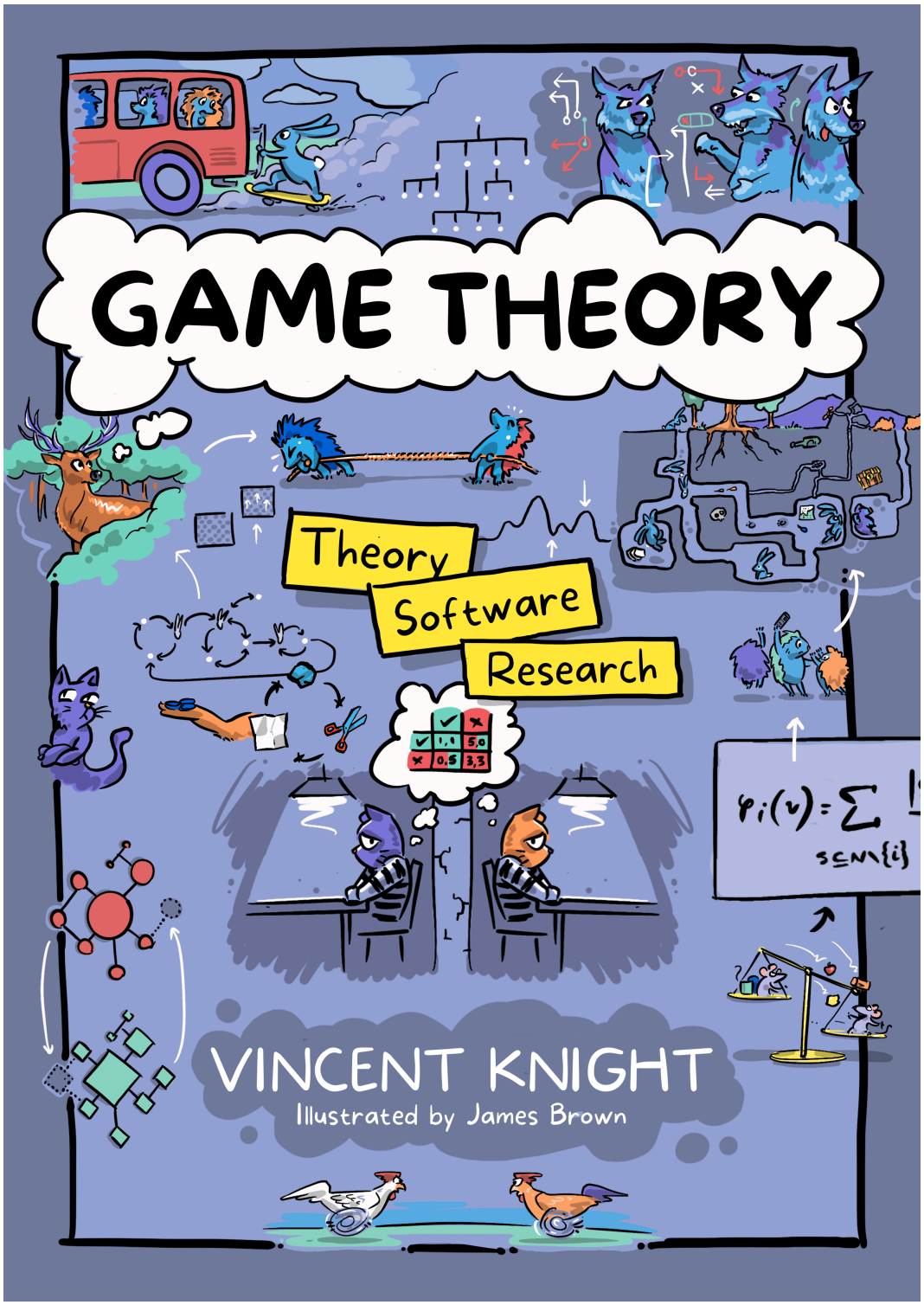
Game Theory


GAME THEORY
Theory
Software
Research
VINCENT KNIGHT
Illustrated by James Brown

Contents

1. Introduction

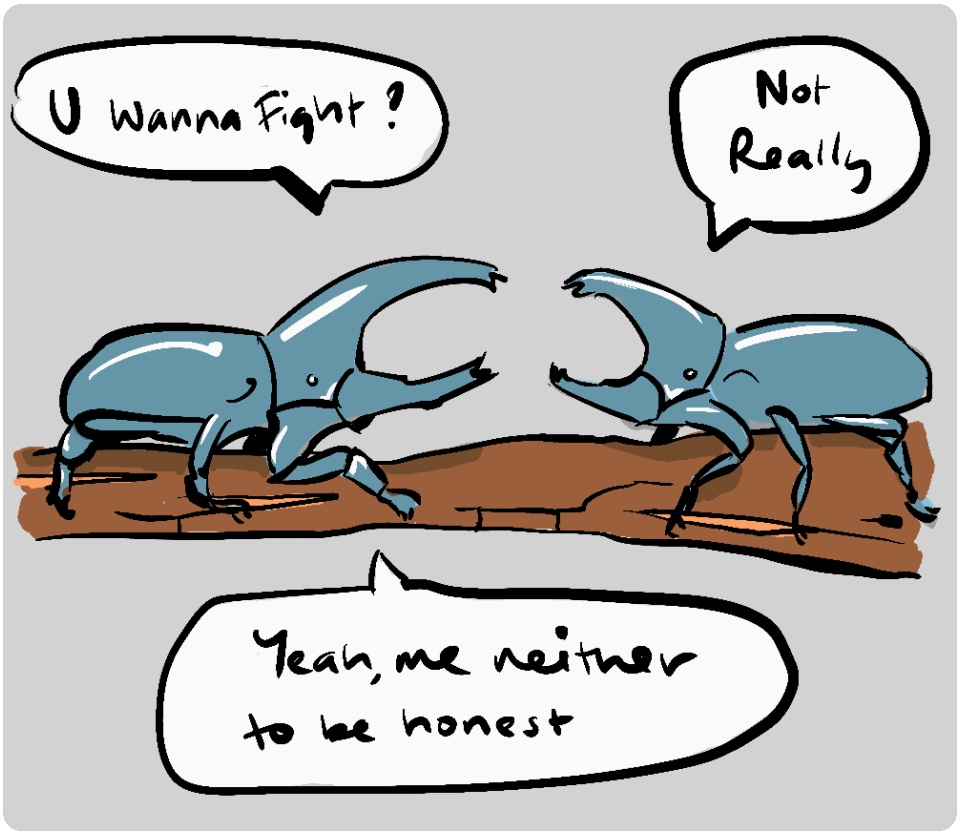


Figure 1.1: Two Hercules beetles weigh up a contest that neither truly wants. The escalation of conflict, and the conditions under which it is or is not worth it, is a recurring theme in game theory.

1.1 What is Game Theory?

A common definition of the subject is:

> "Game theory is the study of interactive decision-making where the outcomes of one decision maker's choices depend on the decisions made by others."

While this definition captures the essence of strategic interaction, game theory is much more than that. It is a beautiful mathematical discipline with deep theoretical avenues for exploration. It plays a crucial role in the global economy, with multiple Nobel Prizes awarded for contributions to our understanding of markets. It provides models that explain how biological structures, from cells to ecosystems, evolve over time. It offers insights into cooperation, competition, and the formation of social norms.

Because game theory spans multiple disciplines, no single definition fully captures its scope. This book aims to provide a solid understanding of game theory: what it can do and how to apply it.

1.2 Who is this book for?

This book is written primarily for advanced undergraduate mathematicians and computer scientists. It may also serve as a starting point for early-career researchers seeking a practical understanding of game theory.

However, the book aims to be accessible to aspiring game theorists from any discipline. A psychologist modelling a specific behaviour? A conservationist analysing the conditions under which a policy is likely to succeed? An economist studying strategic interactions in competitive markets? A

computer programmer implementing a game-theoretic algorithm? Whatever your background, this book provides the necessary tools to engage with game theory in a meaningful way.

The book includes appendices that introduce mathematical theory independently of game theory. For some readers, these may serve as a review of familiar topics, while for others, they offer a first introduction to key techniques needed to apply game-theoretic ideas effectively.

Each chapter includes a section demonstrating how software can be used to apply the ideas at scale. These sections assume some familiarity with Python and the ability to install external libraries. For an introduction to these topics, see the chapters on Using Notebooks and Installing Libraries in the *Python for Mathematics* text.

Game theory is a field that thrives on cross-disciplinary insights, and this book is designed to help readers from different backgrounds develop a shared mathematical foundation. Whether your interest is theoretical or applied, the goal is to equip you with the tools to explore game theory with confidence.

1.3 Licence and access

There will **always be a free online version** of this book, available at vknight.org/gtb. A PDF version is built alongside the site and can be downloaded from vknight.org/gtb/pdf/main.pdf, and the full source is on GitHub at github.com/drvinceknight/gt, from which the site is built and served.

The book brings together material under three sets of terms. The prose, together with the figures and diagrams created by the authors, is released under the Creative Commons Attribution 4.0 International Licence (CC BY 4.0), so it may be shared and adapted with attribution. The source code, including the executable code cells and the build tooling, is released under the MIT Licence.

The illustrations are the work of James Brown (nonzerosum.games) and are treated differently. They remain the copyright of their creator and are included here by permission. They are **not** covered by the CC BY 4.0 licence that applies to the rest of the book: James Brown retains full ownership of the illustrations, may continue to use them and any variants of them in his own work, and any reuse of an illustration outside this book requires his permission. Full details are in the `LICENSE` and `ILLUSTRATIONS.md` files in the repository.

1.4 Citing this book

If this book contributes to your work we would appreciate a citation. A suggested plain-text form is:

> Knight, V. and Brown, J. (2026). *Game Theory: Theory, Software, Research.* Available at vknight.org/gtb.

The corresponding BibTeX entry is:

```
@book{knight2026gametheory,
  author = {Knight, Vincent and Brown, James},
  title  = {Game Theory: Theory, Software, Research},
  year   = {2026},
  url    = {https://vknight.org/gtb/},
}
```

The repository also includes a `CITATION.cff` file, from which GitHub can generate a citation in a number of common formats.

1.5 What does this book cover?

The book is organised into three broad themes.

Foundations of strategic interaction. The opening chapters establish the core language of game theory. The Games chapter introduces normal and extensive form representations, strategies, and utilities. Rationalisation develops the idea of best responses and iterated elimination of dominated strategies. Zero-Sum Games shows how minimax optimisation and linear programming characterise optimal play when players' interests are directly opposed. Nash Equilibrium formalises the central solution concept, a strategy profile from which no player wishes to deviate unilaterally, and the support enumeration algorithm provides a systematic way to compute it. Subgame Perfection refines Nash equilibrium for extensive form games by requiring rationality at every decision node, not just on the equilibrium path.

Dynamics and long-run behaviour. The middle chapters study how equilibria arise and persist over time. Repeated Games examines how cooperation can be sustained when players interact indefinitely, culminating in the Folk Theorem. Direct Reciprocity extends this to memory-one strategies, showing how simple conditional rules such as Tit-for-Tat can stabilise cooperation. Evolutionary Biology provides the wider biological context for evolutionary game theory and motivates the dynamical models that follow; it is optional reading for a purely mathematical pass through the book. Replicator Dynamics models how strategy frequencies evolve in large populations under selection pressure. The Moran Process studies fixation in finite populations. Learning and Evolutionary Dynamics examines how the choice of update rule (imitation, best response, or generational turnover) shapes long-run outcomes across these models. Best Response Polytopes introduces the Lemke–Howson algorithm as a geometric method for computing Nash equilibria in two-player games.

Allocation and collective choice. The final chapters apply game-theoretic reasoning to settings where resources, partners, or decisions must be shared. Routing Games studies how selfish routing decisions lead to inefficiency (the Price of Anarchy). Matching Games covers the Gale–Shapley algorithm and stable matchings. Auction Games analyses first- and second-price auctions and Bayesian equilibrium bidding strategies. Social Choice investigates collective decision-making and impossibility results. Cooperative Games considers coalition formation, the characteristic function, and the Shapley value.

Five **appendices** provide self-contained mathematical background: numerical integration, absorbing Markov chains, ergodic Markov chains, interior point optimisation (KKT conditions), and integer pivoting.

Throughout, the emphasis is on games of complete information, in which the players, their available actions, and their payoffs are common knowledge. This is the classical setting in which the core solution concepts are cleanest to state and compute. Games of incomplete information, where players hold private information about their own payoffs, enter only in the Auction Games chapter, through the notion of a Bayesian Nash equilibrium among bidders with private valuations. A systematic treatment of Bayesian games, mechanism design, and signalling lies beyond the scope of this book, and is a natural direction for further study; the auctions chapter is intended as a first point of contact with these ideas rather than a complete account of them.

1.6 How is this book different from similar books?

There are a number of excellent books on game theory that are highly recommended. For a fantastic introduction to the topic aimed at a mathematical audience, see (Webb, 2007). The exhaustive work (Maschler et al., 2020) offers a vast amount of breadth and depth on the subject, while (Roughgarden, 2010) delves into modern algorithmic approaches to modelling complex systems at an advanced level.

Some excellent texts focus on specific subtopics within the domain. For example, (Gusfield & Irving, 1989) explores matching games, and (Roughgarden, 2002) addresses routing games. Other great

general introductions include (Osborne & others, 2004), (Watson, 2002), and (Gusfield & Irving, 1989), to name just a few.

This book, however, offers something that these fantastic works do not: detailed implementation instructions, including multiple examples and exercises that show how to solve specific problems, even when they are large and complex. It also provides an overview of open-source software tools that are readily available to solve real-world problems. The book also includes discussion of relevant contemporary research aiming to demonstrate not only how the topic is applied in practice but also its relevance. Finally, it offers a rigorous theoretical foundation, serving as a springboard for deeper theoretical analysis.

1.7 How is this book organised?

Each chapter in this book follows a structured format:

1. **A motivating case study:** a running example throughout the chapter to illustrate and contextualise the theory, often drawing from real-world scenarios.
2. **Relevant definitions and theory:** providing a rigorous mathematical foundation for the chapter's topic.
3. **Exercises:** a range of problems inviting the reader to apply the definitions and theory to deepen understanding.
4. **Practical implementations:** including Python code, and an overview of relevant open-source tools to guide implementation.
5. **Contemporary research:** a hand-picked, non-exhaustive overview of relevant research related to the chapter.

Some readers may prefer to focus on the theoretical foundations, while others might engage primarily with the examples and practical implementations. This book is designed to be flexible, allowing readers to approach the material in a way that best suits their background and interests.

Figure 1.2 shows the general structure of the book and how the various chapters link together.

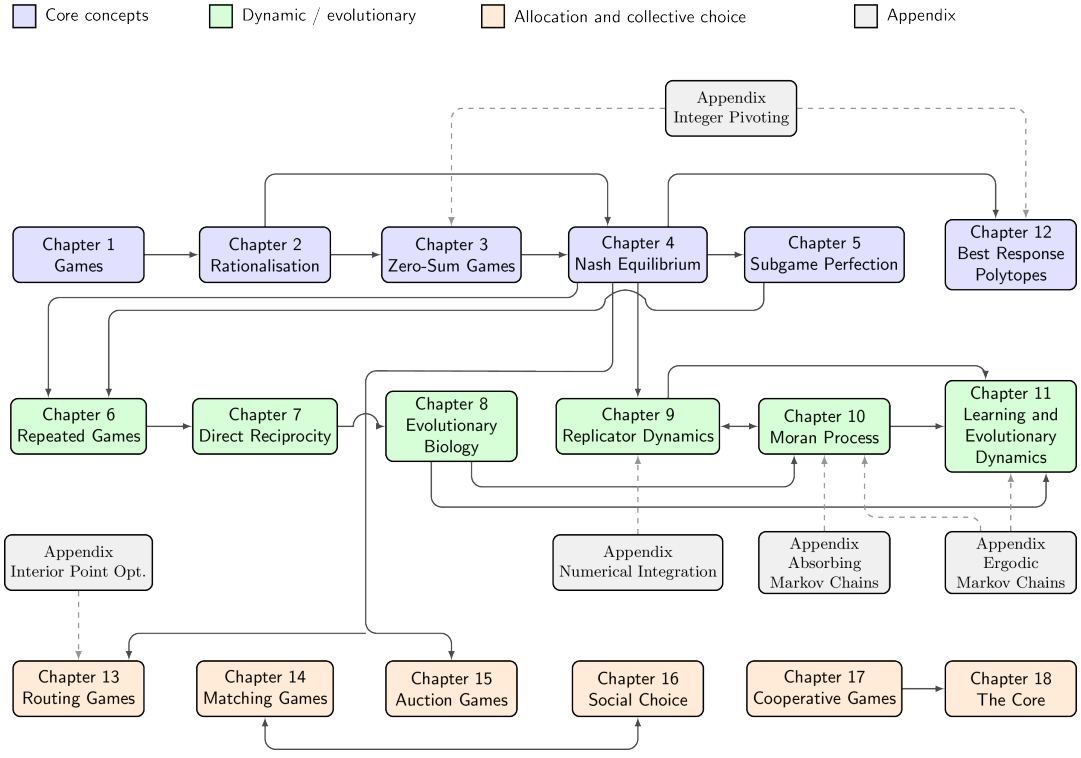


Figure 1.2: Visual representation of the relationships and flow between chapters.

2. Games

Game theory begins with a precise mathematical language for describing strategic interaction. This chapter introduces the two fundamental representations, normal form and extensive form, along with the core concepts of strategies and utilities that underpin everything that follows.

2.1 Motivating Example

Consider the following scenario: you would like to meet your **friends** for coffee. You know the following about their behaviour:

- With probability 20%, they choose coffee house A.
- With probability 80%, they choose coffee house B.

If your goal is to maximise the probability of meeting them, what should you do?

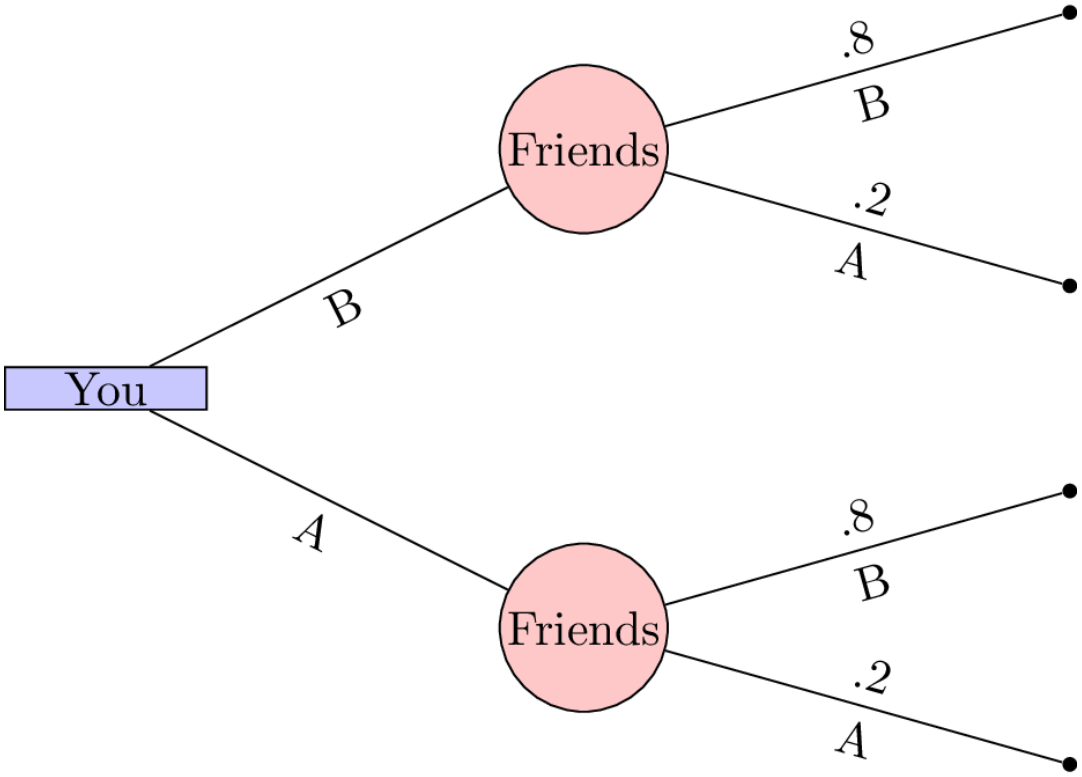


Figure 2.1: A probabilistic decision tree. A square node is used to indicate a decision and a circular node is used to indicate a probabilistic event.

As illustrated in Figure 2.1, if you must choose your location without communication, the optimal choice is coffee house B, offering an 80% chance of meeting your friends.

However, in many situations, decisions are not made in isolation. Often, one party moves first, and others respond after observing that choice. If you were able to announce your decision before your friends chose where to go, the interaction would become a sequential decision problem, an instance of a **game**. Figure 2.2 shows this game.

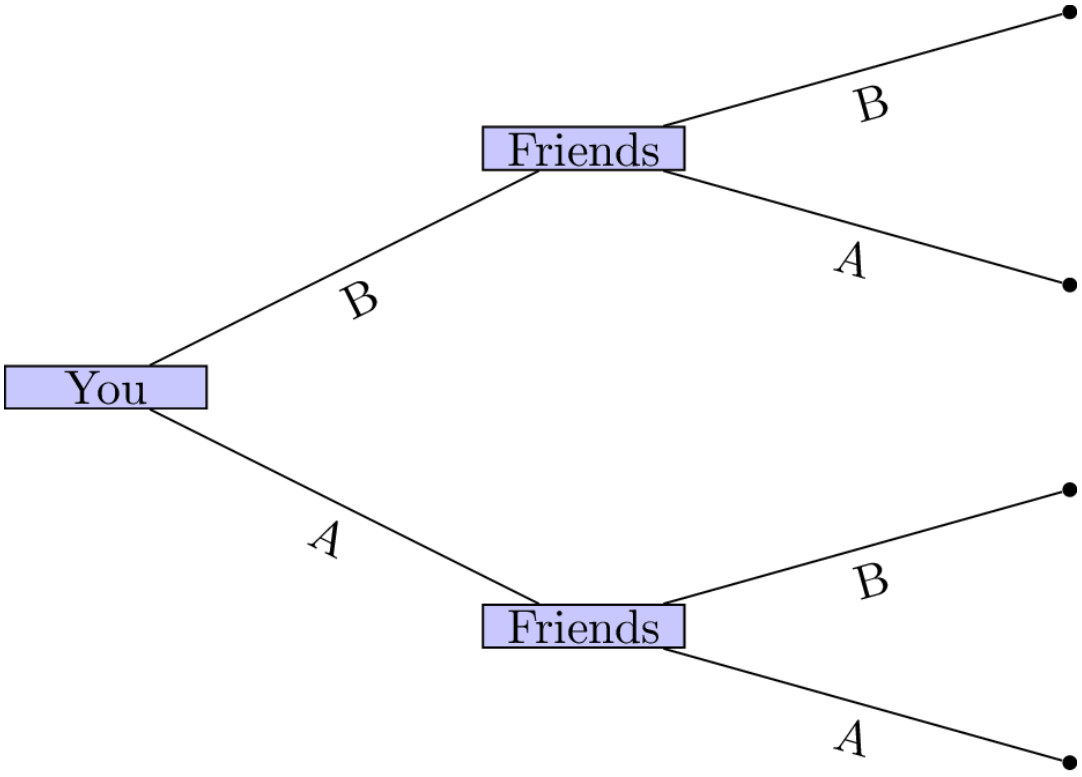


Figure 2.2: Visual representation of the consecutive decisions.

In such games, outcomes depend not only on chance but critically on the sequence of actions taken by all players.

2.2 Theory

In game theory, trees such as Figure 2.2 are used to represent a particular type of game known as an **extensive form game**.

2.2.1 Definition: Extensive Form Game

An N-player extensive form game **with complete information** consists of:

- A finite set of players $\mathcal{N}$ with $|\mathcal{N}| = N$.
- A tree $G = (V, E, x^0)$ where:
  - V is the set of vertices,
  - E is the set of edges, and
  - $x^0 \in V$ is the root of the tree.
- A partition $(V_i)_{i \in \mathcal{N}}$ of the non-terminal vertices, assigning each decision node to a player.
- A set O of possible outcomes.
- A function u mapping each terminal node (leaf) of G to an element of O.

2.2.1.1 Example: Sequential Coordination Game

Consider the following game, often referred to as the **Battle of the Sexes**.

Two friends must decide which movie to watch at the cinema.
Bob prefers a comedy, while Celine prefers a sports movie.
However, both would rather attend the cinema together than go separately.

The structure of the game and the utilities for Bob and Celine are shown in Figure 2.3.

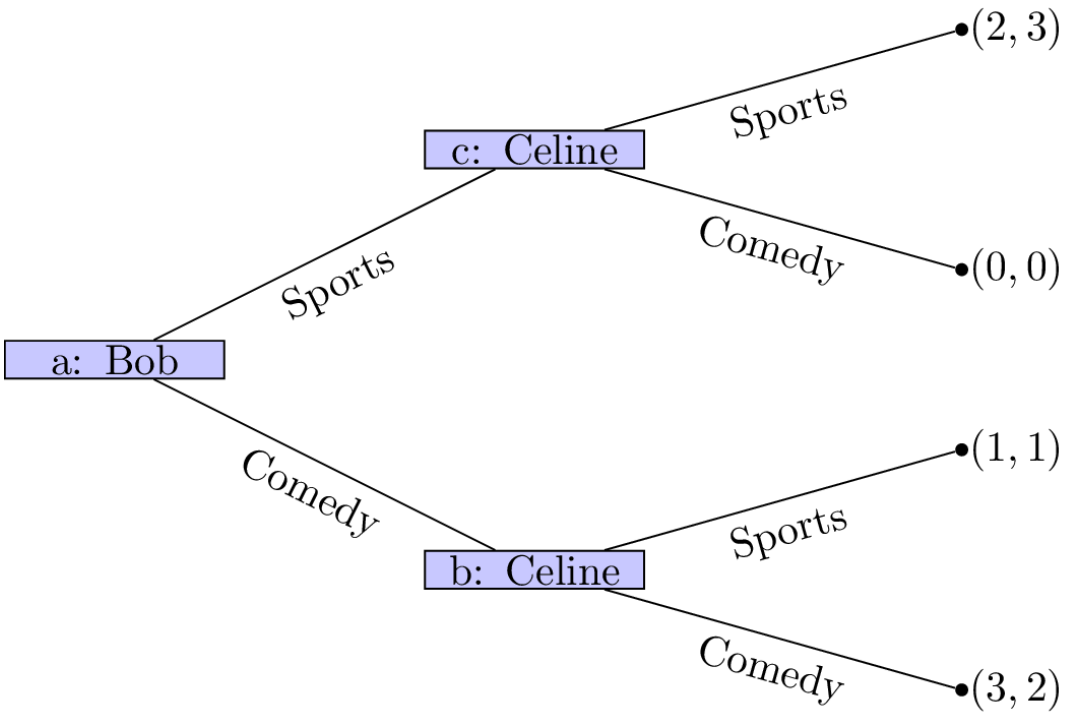


Figure 2.3: The coordination game with Bob choosing first.

If Bob chooses first, the outcome can be predicted as follows:

1. Bob observes that whatever choice he makes, Celine will prefer to follow and select the same type of movie.
2. Bob, anticipating this, chooses a comedy to secure the slightly higher utility he prefers.

This reasoning uses a method known as **backward induction**, which we will formalise in Subgame Perfection.

Alternatively, we can model the game with Celine moving first, as shown in Figure 2.4.

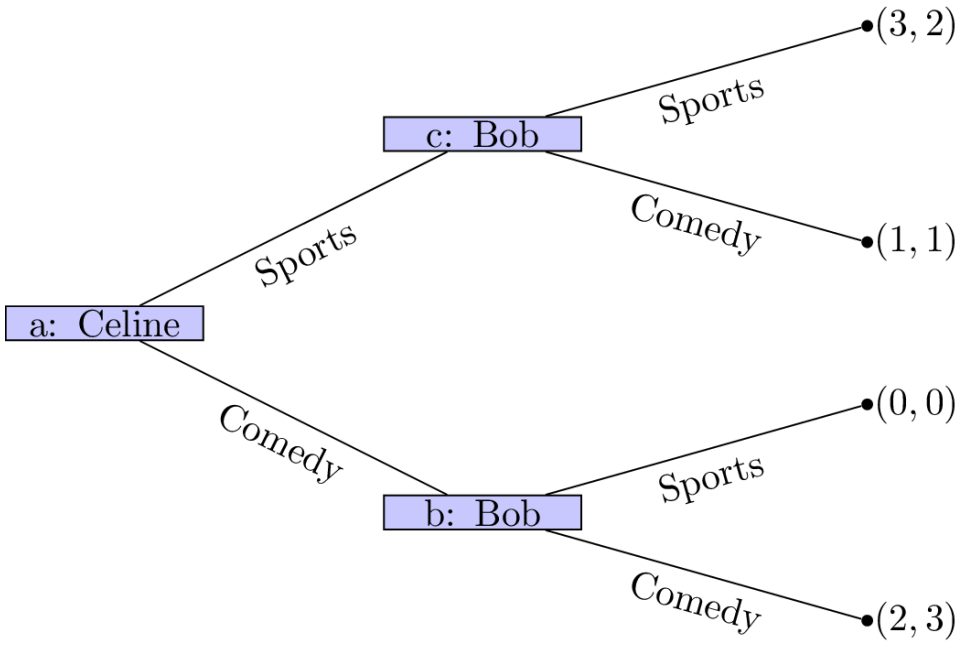


Figure 2.4: The coordination game with Celine choosing first.

In this version:

1. Celine observes that whatever choice she makes, Bob will prefer to follow and select the same type of movie.
2. Celine, anticipating this, chooses the sports movie to secure her preferred outcome.

In both representations, we assume that players have complete information about previous actions when making their decisions. Specifically, we assume that the information available at nodes **b** and **c** differs: players know precisely where they are in the game. This assumption will later be relaxed when we study games with **imperfect information**.

2.2.2 Definition: Information Set

Given a game in extensive form $\left(\mathcal{N}, G, (V_i)_{i\in\mathcal{N}}, O, u\right)$,
the set of information sets v_i for player $i \in \mathcal{N}$ is a partition of V_i.

Each element of v_i denotes a set of nodes at which the player cannot distinguish their exact location when choosing an action.

Two nodes of a game tree are said to belong to the same **information set** if the player making a decision at that point cannot distinguish between them.

Attention

This implies that:

- Every information set contains vertices for a single player.
- All vertices in an information set must have the same number of successors (with the same action labels).

We represent nodes that are part of the same information set diagrammatically by connecting them with a dashed line, as shown in Figure 2.5.

In our example involving Celine and Bob, if both players must choose a movie without knowing the other's choice, nodes **b** and **c** belong to the same information set.

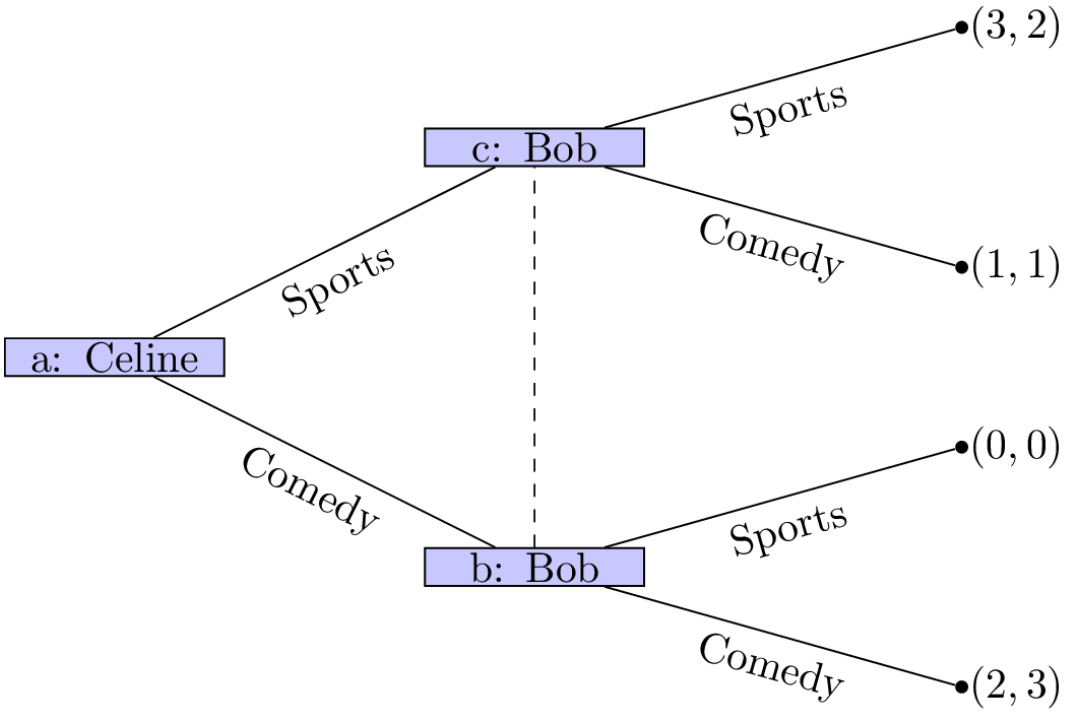


Figure 2.5: The coordination game with imperfect information.

In this case, it becomes significantly less trivial to predict the outcome of the game.

Another way to represent a game is in **normal form**.

2.2.3 Definition: Normal Form Game

An N-player normal form game consists of:

- A finite set of N players.
- An action set for the players: $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_N\}$.
- A set of payoff functions for the players: $u_i : \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_N \to \mathbb{R}$.

Unless otherwise stated, we assume throughout this text that all players act to maximise their individual payoffs.

2.2.3.1 Bi-Matrix Representation

One common way to represent a two-player normal form game is using a **bi-matrix**. Suppose that $N = 2$ and the action sets are: $\mathcal{A}_1 = \{r_i \texttt{ mid } 1 \leq i \leq m\}$ and $\mathcal{A}_2 = \{c_j \texttt{ mid } 1 \leq j \leq n\}$.

A general bi-matrix mapping rows (actions of the first player) and columns (actions of the second player) to pairs of payoff values:

$$\Gamma = \begin{pmatrix} (u_1(r_1, c_1), u_2(r_1, c_1)) & (u_1(r_1, c_2), u_2(r_1, c_2)) & ... & (u_1(r_1, c_n), u_2(r_1, c_n)) \\ (u_1(r_2, c_1), u_2(r_2, c_1)) & (u_1(r_2, c_2), u_2(r_2, c_2)) & ... & (u_1(r_2, c_n), u_2(r_2, c_n)) \\ \vdots & ... & ... & \vdots \\ (u_1(r_m, c_1), u_2(r_m, c_1)) & (u_1(r_m, c_2), u_2(r_m, c_2)) & ... & (u_1(r_m, c_n), u_2(r_m, c_n)) \end{pmatrix} \tag{2.1}$$

An alternative approach is to use two separate matrices: M_r for the row player and M_c for the column player, defined as follows:

$$(M_r)_{ij} = u_1(r_i, c_j) \qquad (M_c)_{ij} = u_2(r_i, c_j) \tag{2.2}$$

This is the preferred representation used in this text.

2.2.3.2 Example: Coordination game

The Normal Form Game representation of Figure 2.5 has the following payoff matrices for the row and column players:

$$M_r = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix} \tag{2.3}$$

2.2.3.3 Example: Prisoners' Dilemma

Two thieves have been caught by the police and separated for questioning. If both cooperate (remain silent), they each receive a short sentence. If one defects (betrays the other), they are offered a deal while the other receives a long sentence. If both defect, they both receive a medium-length sentence.

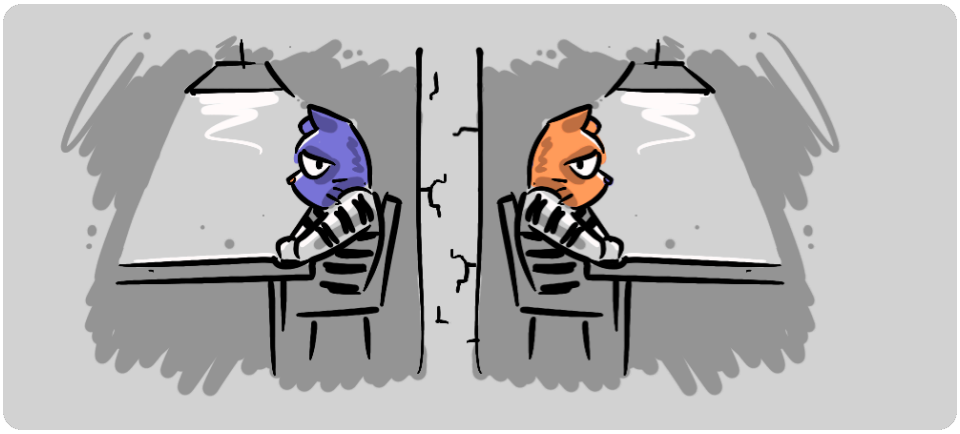


Figure 2.6: Two suspects, questioned separately, each decide whether to stay silent or betray the other. The Prisoners' Dilemma is the canonical example of a game in which individually rational play leaves both players worse off.

The normal form of this game is represented by:

$$M_r = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{2.4}$$

2.2.3.4 Example: Hawk–Dove

Suppose two birds of prey must share a limited resource. Hawks always fight over the resource, potentially to the death or at great cost, and dominate doves. Doves, on the other hand, avoid conflict and share the resource if paired with another dove.

This interaction is modelled as:

$$M_r = \begin{pmatrix} 0 & 3 \\ 1 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix} \tag{2.5}$$

2.2.3.5 Example: Pigs

Two pigs, one dominant and one subservient, share a pen containing a lever that dispenses food. Pressing the lever takes time, giving the other pig an opportunity to eat. If the dominant pig presses the lever, the subservient pig eats most of the food before being pushed away. If the subservient pig presses the lever, the dominant pig eats all the food. If both pigs press the lever, the subservient pig manages to eat a third of the food.

The game is represented by:

$$M_r = \begin{pmatrix} 4 & 2 \\ 6 & 0 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 3 \\ -1 & 0 \end{pmatrix} \tag{2.6}$$

2.2.3.6 Example: Matching Pennies

Two players simultaneously choose to display a coin as either Heads or Tails.
If both players show the same face, player 1 wins.
If the faces differ, player 2 wins.

This zero-sum game is represented as:

$$M_r = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.7}$$

We now discuss how players choose their actions in both normal and extensive form games.

2.2.4 Definition: Strategy in Normal Form Games

A strategy for a player with action set $\mathcal{A}$ is a probability distribution over the elements of $\mathcal{A}$.

Typically, a strategy is denoted by $\sigma \in [0,1]_{\mathbb{R}}^{|\mathcal{A}|}$, subject to the condition:

$$\sum_{i=1}^{|\mathcal{A}|} \sigma_i = 1 \tag{2.8}$$

Given an action set $\mathcal{A}$ the set of valid strategies is denoted as $\Delta\mathcal{A}$ so that:

$$\Delta\mathcal{A} = \left\{ \sigma \in [0,1]_{\mathbb{R}}^{|\mathcal{A}|} \mid \sum_{i=1}^{|\mathcal{A}|} \sigma_i = 1 \right\} \tag{2.9}$$

In the Matching Pennies game, a strategy profile such as
$\sigma_1 = (0.2, 0.8)$ and $\sigma_2 = (0.6, 0.4)$ implies that player 1 plays Heads with probability 0.2, and player 2 plays Heads with probability 0.6.

We can extend the utility function, which maps from action profiles to $\mathbb{R}$, using **expected utility**. For a two-player game, we write:

$$u_i(\sigma_1, \sigma_2) = \sum_{a_1 \in \mathcal{A}_1} \sum_{a_2 \in \mathcal{A}_2} \sigma_1(a_1)\sigma_2(a_2)u_i(a_1, a_2) \tag{2.10}$$

Here, we treat σ_i as a function $\sigma_i : \mathcal{A}_i \to [0,1]$ so that $\sigma_i(a_i)$ is the probability of choosing action $a_i \in \mathcal{A}_i$.

Warning

In a lot of game theoretic texts strategies in the context of Normal Form games are referred to as **Mixed Strategies** and strategies that play a single action are referred to as **Pure Strategies**.

2.2.4.1 Example: Expected Utility in Matching Pennies

Using the Matching Pennies game and the strategy profile $\sigma_1 = (0.2, 0.8)$ and $\sigma_2 = (0.6, 0.4)$, the expected utilities are:

$$u_1(\sigma_1, \sigma_2) = 0.2 \cdot 0.6 \cdot 1 + 0.2 \cdot 0.4 \cdot (-1) + 0.8 \cdot 0.6 \cdot (-1) + 0.8 \cdot 0.4 \cdot 1 = -0.12 \quad (2.11)$$

$$u_2(\sigma_1, \sigma_2) = 0.2 \cdot 0.6 \cdot (-1) + 0.2 \cdot 0.4 \cdot 1 + 0.8 \cdot 0.6 \cdot 1 + 0.8 \cdot 0.4 \cdot (-1) = 0.12 \quad (2.12)$$

Now, suppose player 2 always plays Tails. Then $\sigma_2 = (0, 1)$, and if $\sigma_1 = (x, 1 - x)$, we have:

$$u_1(\sigma_1, \sigma_2) = x \cdot -1 + (1 - x) \cdot 1 = 1 - 2x \quad (2.13)$$

Similarly, if player 1 always plays Tails, and $\sigma_2 = (y, 1 - y)$, the expected utility for player 2 is:

$$u_2(\sigma_1, \sigma_2) = x \cdot y \cdot (-1) + x \cdot (1 - y) \cdot 1 + (1 - x) \cdot y \cdot 1 + (1 - x) \cdot (1 - y) \cdot (-1) \quad (2.14)$$

If we simplify and assume player 1 always plays Tails, then $x = 0$ and we get:

$$u_2(\sigma_1, \sigma_2) = y \cdot 1 + (1 - y) \cdot (-1) = 2y - 1 \quad (2.15)$$

We can visualise both of these expected utility functions in Figure 2.7.

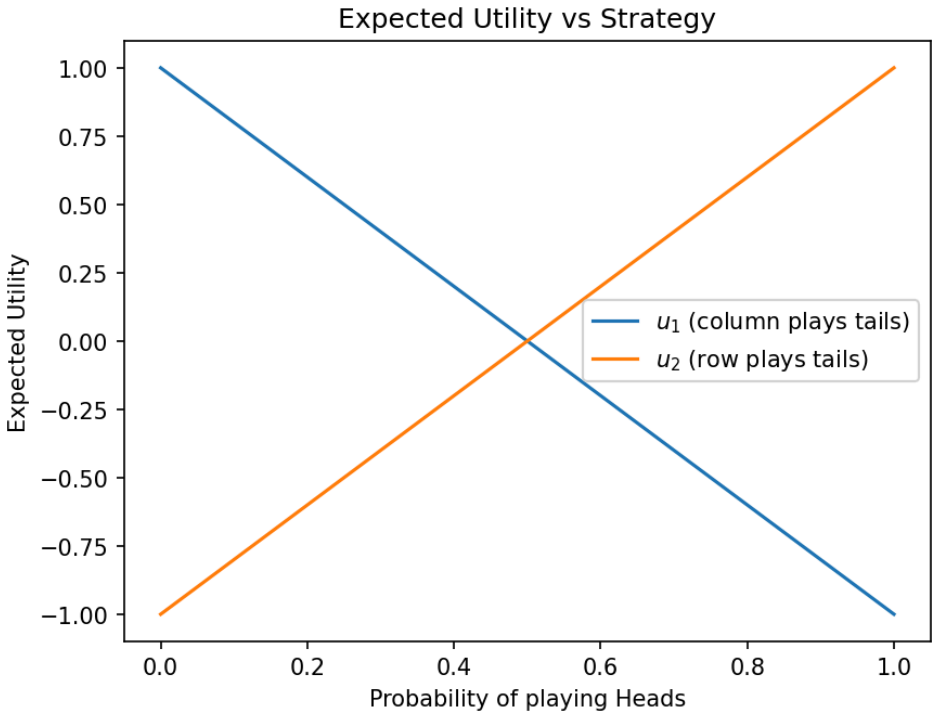


Figure 2.7: Expected utility for each player as a function of the probability of playing Heads in Matching Pennies.

2.2.5 Definition: Strategy in Extensive Form Games

A strategy for a player in an Extensive Form Game is a mapping from each information set to a probability distribution over the available actions at that set.

2.2.5.1 Example: Strategies in the Sequential Coordination Game

In the Figure 2.3, Bob has a single decision node. His strategy can be represented as:

$$a \to (x, 1-x) \tag{2.16}$$

where x is the probability of choosing "sports".

Celine has two decision nodes, which depend on Bob's choice. Her strategy is:

$$\begin{aligned} b &\to (x_b, 1-x_b) \\ c &\to (x_c, 1-x_c) \end{aligned} \tag{2.17}$$

Here, x_b is the probability of choosing "sports" if Bob chose sports, and x_c is the probability of choosing sports if Bob chose comedy.

If Celine is unaware of Bob's decision, as in the Figure 2.5, then nodes **b** and **c** form a single information set. Her strategy becomes:

$$\{b, c\} \to (x, 1-x) \tag{2.18}$$

This leads us to a fundamental idea: **every extensive form game corresponds to a normal form game**, where strategies describe complete plans of action across all information sets.

2.2.6 Mapping Extensive Form Games to Normal Form

An extensive form game can be mapped to a normal form game by enumerating all possible strategies that specify a single action at each information set. This collection of strategies defines the action sets in the corresponding normal form game.

These strategies can be thought of as vectors in the Cartesian product of the action sets available at each information set. For a player $i \in \mathcal{N}$ with information sets $v_i = \left((v_i)_1, (v_i)_2, ..., (v_i)_n\right)$, a strategy $s = (s_1, s_2, ..., s_n)$ prescribes one action at each information set.
For example, s_2 specifies the action to take at every vertex contained within $(v_i)_2$.

2.2.6.1 Example: Strategy Enumeration in the Sequential Coordination Game

Consider the Figure 2.3. We enumerate all strategies for each player by listing single actions taken at each information set.

For Bob, who has a single information set, the list of all strategies in the extensive form game that prescribe a single action is:

$$\left\{ a \to (1,0), \quad a \to (0,1) \right\} \tag{2.19}$$

These correspond to always choosing **Sports** or always choosing **Comedy**. The action set in the Normal Form Game is:

$$\mathcal{A}_1 = \{\text{Sports},\ \text{Comedy}\} \tag{2.20}$$

For Celine, who has two information sets (after Bob's move), the pure strategies in the extensive form are:

$$
\begin{aligned}
&(b \to (1,0),\, c \to (1,0)) \\
&(b \to (1,0),\, c \to (0,1)) \\
&(b \to (0,1),\, c \to (1,0)) \\
&(b \to (0,1),\, c \to (0,1))
\end{aligned}
\tag{2.21}
$$

These strategies describe how Celine responds to each of Bob's possible choices. Thinking of these as vectors in the Cartesian product of available actions at each information set, we can define Celine's action set in the Normal Form Game as:

$$\mathcal{A}_2 = \{(\text{Sports}, \text{Sports}), (\text{Sports}, \text{Comedy}), (\text{Comedy}, \text{Sports}), (\text{Comedy}, \text{Comedy})\} \tag{2.22}$$

Using these enumerated strategies, the payoff matrices for Bob (the row player) and Celine (the column player) are:

$$M_r = \begin{pmatrix} 3 & 3 & 1 & 1 \\ 0 & 2 & 0 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 2 & 1 & 1 \\ 0 & 3 & 0 & 3 \end{pmatrix} \tag{2.23}$$

2.3 Exercises

Exercise 2.1:

Using Figure 2.3 and the definition of an Extensive Form Game:

1. What is the finite set of players $\mathcal{N}$?
2. What are the elements of the game tree $G = (V, E, x^0)$?
3. What is the partition $(V_i)_{i \in \mathcal{N}}$?
4. What is the set of possible game outcomes O?
5. What is the mapping u from every leaf of G to an element of O?

Exercise 2.2:

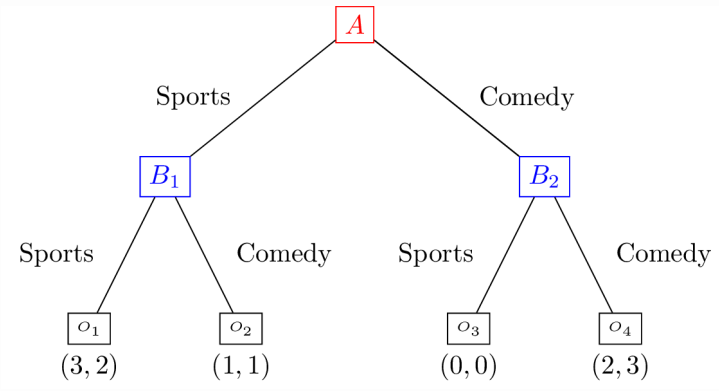
Using Figure 2.5 and the definition of an Extensive Form Game:

1. What is the finite set of players $\mathcal{N}$?
2. What are the elements of the game tree $G = (V, E, x^0)$?
3. What is the partition $(V_i)_{i \in \mathcal{N}}$?
4. What is the set of possible game outcomes O?
5. What is the mapping u from every leaf of G to an element of O?

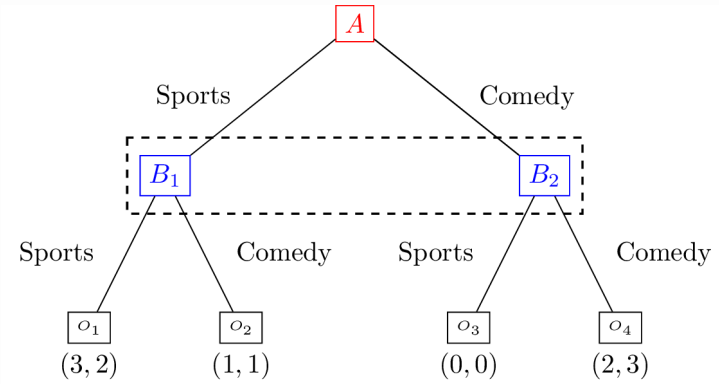
Exercise 2.3:

For each of the following games with $\mathcal{N} = \{\text{Alice}, \text{ Bob}\}$, assume that decision nodes A_i belong to Alice and B_i belong to Bob. Determine all **information sets**.

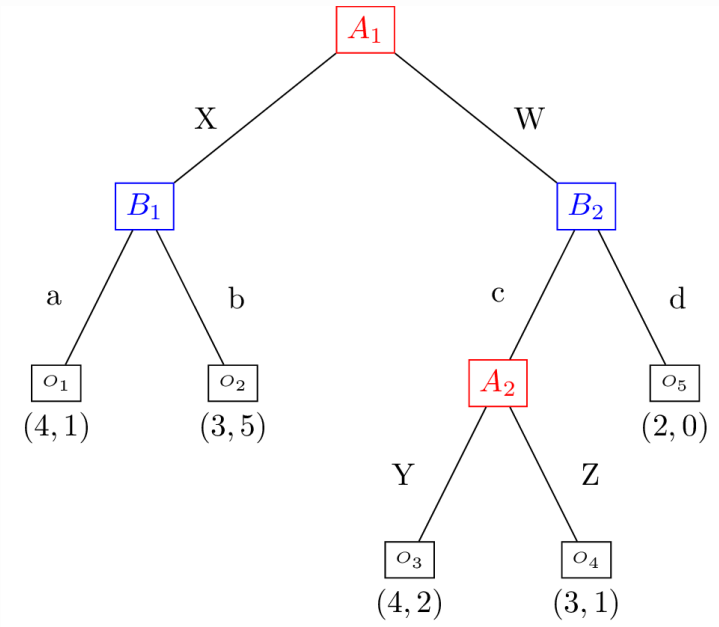
1.


A
Sports
Comedy
B1
B2
Sports
Comedy
Sports
Comedy
o1
o2
o3
o4
(3, 2)
(1, 1)
(0, 0)
(2, 3)

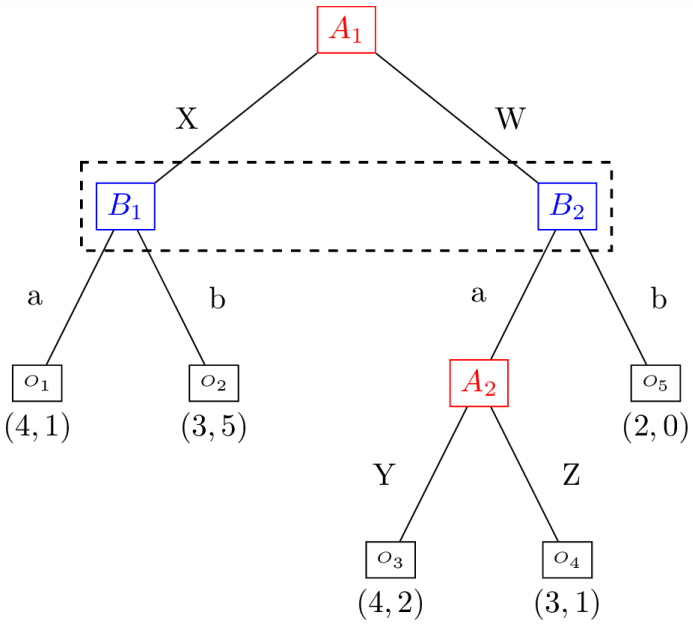
2.


A
Sports
Comedy
B1
B2
Sports
Comedy
Sports
Comedy
o1
o2
o3
o4
(3, 2)
(1, 1)
(0, 0)
(2, 3)

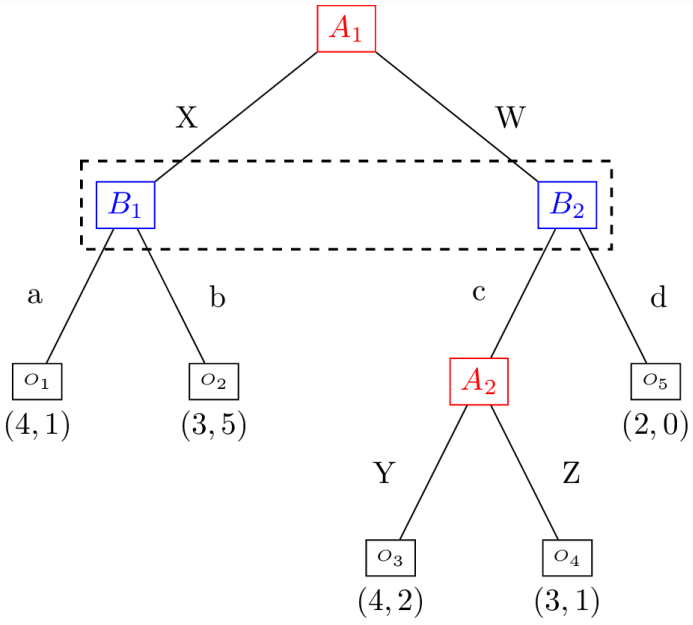
3.


A1
X
W
B1
B2
a
b
c
d
o1
o2
A2
o5
(4, 1)
(3, 5)
(2, 0)
Y
Z
o3
o4
(4, 2)
(3, 1)

4.



5.



Exercise 2.4:

Alice, Bob, and Celine are childhood friends who want to stay in touch online. Each has a choice between three messaging platforms: **WhatsApp**, **Instagram**, and **Snapchat**.

- Clearly identify the players and their strategy sets.
- Propose a reasonable interpretation of utility values.
- Write the normal form representation of the game.

Exercise 2.5:

Two neighbouring countries possess highly destructive military forces.

- If both attack, each suffers **10,000** civilian casualties.
- If one attacks while the other remains peaceful, the peaceful country suffers **15,000** casualties while the attacker suffers **13,000** in retaliation.
- If both remain peaceful, there are **no** casualties.

- Clearly state the players and their strategy sets.
- Plot the expected utility to each country assuming it plays a strategy while the other remains peaceful.
- Plot the expected utility to each country assuming it plays a strategy while the other attacks.

2.4 Programming

2.4.1 Linear Algebraic Form of Expected Utility

The expected utility calculation defined in (2.10) corresponds to a matrix formulation using linear algebra:

$$u_i(\sigma_1, \sigma_2) = \sigma_1 M \sigma_2^T \tag{2.24}$$

Here, M is the payoff matrix for player i, and σ_1, σ_2 are row and column player strategies, treated as row vectors.

This linear algebraic form will be important in later chapters, and it also enables **efficient numerical computation** in programming languages that support vectorized matrix operations.

2.4.2 Using NumPy to Compute Expected Utilities

Python's numerical library **NumPy** (Harris et al., 2020) provides vectorized operations through its `array` class. Below, we define the elements from the earlier Matching Pennies example:

```
import numpy as np

M_r = np.array(
    (
        (1, -1),
        (-1, 1),
    )
)
M_c = -M_r

sigma_r = np.array((0.2, 0.8))
sigma_c = np.array((0.6, 0.4))
```

Note

We use the fact that $M_r = -M_c$ here, which holds because the Matching Pennies game happens to be a **zero-sum game**, a topic discussed in the Zero-Sum Games chapter.

We can now compute the expected utility for the row player using (2.24):

```
print(f"Row player expected utility: {sigma_r @ M_r @ sigma_c}")
```

```
Row player expected utility: -0.12
```

The equivalent calculation for the column player is:

```
print(f"Column player expected utility: {sigma_r @ M_c @ sigma_c}")
```

```
Column player expected utility: 0.12
```

Important

Use `@` for matrix multiplication, not `*`, which performs element-wise multiplication.

2.4.3 Using Nashpy to Create Games and Compute Utilities

The Python library **Nashpy** (Knight & Campbell, 2018) is specifically designed for two-player normal form games. We can use it to create the Matching Pennies game:

```
import nashpy as nash

matching_pennies = nash.Game(M_r, M_c)
print(matching_pennies)
```

```
Zero sum game with payoff matrices:

Row player:
[[ 1 -1]
 [-1  1]]

Column player:
[[-1  1]
 [ 1 -1]]
```

Note

We reuse `M_r` and `M_c` from Using NumPy to Compute Expected Utilities and so do not need to redefine them.

To compute the expected utilities of both players:

```
print(f"Expected utilities (row, column): {matching_pennies[sigma_r, sigma_c]}")
```

```
Expected utilities (row, column): [-0.12  0.12]
```

Note

Nashpy is the main Python library used in this text for studying games. **Gambit** is another Python library that includes implementation of algorithms for games with a larger number of players.

2.4.4 Creating a 3 player game using Gambit

Here we create a Normal Form game in Gambit (Savani & Turocy, 2024) with 3 players:

```
import pygambit as gbt

g = gbt.Game.new_table([2, 2, 2], title="3 player game")
p1, p2, p3 = g.players
```

Now let us write all the utilities of each outcome:

```
g[(0, 0, 0)][p1] = 1
g[(0, 0, 0)][p2] = 1
g[(0, 0, 0)][p3] = 1

g[(0, 0, 1)][p1] = 2
g[(0, 0, 1)][p2] = 1
g[(0, 0, 1)][p3] = -1

g[(0, 1, 0)][p1] = 2
g[(0, 1, 0)][p2] = 1
g[(0, 1, 0)][p3] = -1

g[(0, 1, 1)][p1] = 2
g[(0, 1, 1)][p2] = 1
g[(0, 1, 1)][p3] = -1

g[(1, 0, 0)][p1] = 2
g[(1, 0, 0)][p2] = 1
g[(1, 0, 0)][p3] = -1

g[(1, 0, 1)][p1] = 2
g[(1, 0, 1)][p2] = 1
g[(1, 0, 1)][p3] = -1

g[(1, 1, 0)][p1] = 2
g[(1, 1, 0)][p2] = 1
g[(1, 1, 0)][p3] = -1

g[(1, 1, 1)][p1] = 1
g[(1, 1, 1)][p2] = 1
g[(1, 1, 1)][p3] = 1
print(g)
```

```
Game(title='3 player game')
```

2.5 Notable Research

This chapter has introduced the mathematical foundations needed to define games and strategic behaviour in interactive decision-making. In later chapters, we will analyse **emergent behaviour**, a central focus of much contemporary game theory research.

This section presents a brief overview of selected studies that demonstrate how game theory is applied across a range of real-world scenarios. We highlight the types of situations being modelled rather than focusing on technical details.

In (Lee & Roberts, 2016), the authors develop a game-theoretic model of **rhino horn devaluation**, a conservation strategy in which a rhinoceros's horn is partially removed to reduce its value to

poachers. This work, along with follow-up research such as (Glynatsi et al., 2018), explores the conditions under which poachers are deterred from targeting these endangered animals.

Game theory has also been applied to **healthcare systems**. In (Knight et al., 2017), hospitals are modelled as agents who may strategically limit bed availability to meet performance targets. A related study, (Panayides et al., 2023), examines the interaction between hospitals and ambulance services. Both models use game theory to identify conditions under which rational decision-making may benefit patients overall.

In a different health-related setting, (Basanta et al., 2008) presents a game-theoretic model of **cancer progression**, interpreting the dynamics between different cell types through the lens of evolutionary games.

In (Jordan et al., 2009), the authors model **play calling in American Football**. Their framework incorporates factors such as game stage and risk aversion to identify advantageous strategic decisions in a Normal Form setting.

The study (Kwak et al., 2020) constructs a game-theoretic version of the **Volunteer's Dilemma**, in which individuals can choose to help or remain bystanders. The authors formalise this as a general N-player Normal Form Game and investigate how helping behaviour varies with the cost of intervention.

A related dilemma is explored in (Weitz et al., 2016), where the focus shifts from individual help to shared environmental resources. This model examines how agent behaviour affects the environment, which in turn feeds back to influence future strategic choices.

While the above studies are primarily based on **Normal Form Games**, (Boyd et al., 2023) presents a model in **Extensive Form**. The authors consider a water resource management problem where three firms are located sequentially along a river basin. The model captures the asymmetric benefits faced by upstream versus downstream players. The proposed mechanism, a non-cooperative water-release market, enables upstream agents to release water, creating a more equitable outcome for all parties.

2.6 Conclusion

Table 2.1 compares the two representations covered in this chapter.

Concept	Normal Form Game	Extensive Form Game
Representation	Payoff Matrices	Game tree
Sequence of Moves	Simultaneous	Sequential (possibly with information sets)
Players	$\mathcal{N}$	$\mathcal{N}$
Actions	$\mathcal{A}_i$ per player	Specified at each decision node
Strategies	Probability distributions over actions	Mappings from information sets to probability distributions over actions
Utilities	$u_i(\vec{a})$ for action profiles	$u(x)$ for each terminal node
Information	Perfect (assumes all actions known at once)	Can be perfect or imperfect (via information sets)

Table 2.1: The main concepts for Normal Form and Extensive Form Games.

Attention

Every extensive form game has an equivalent representation in normal form, where strategies specify complete plans of action across all information sets.

2.7 Solutions

Solution 2.1: Solution to Exercise 1

We refer to the Figure 2.3 in which Bob moves first and Celine responds after observing Bob's choice.

1. The finite set of players is:

$$\mathcal{N} = \{\text{Bob},\ \text{Celine}\} \tag{2.25}$$

so $|\mathcal{N}| = 2$.

2. The game tree $G = (V, E, x^0)$ consists of:

- **Vertices** V: the root node a (Bob's decision), two intermediate nodes b and c (Celine's decisions after Bob plays Sports or Comedy respectively), and four terminal (leaf) nodes corresponding to the four possible outcomes.
- **Edges** E: the edge from a to b (labelled Comedy), the edge from a to c (labelled Sports), and the four edges from b and c to the terminal nodes (each labelled Sports or Comedy).
- **Root** $x^0 = a$.

3. The partition $(V_i)_{i \in \mathcal{N}}$ assigns each non-terminal vertex to a player:

$$V_{\text{Bob}} = \{a\}, \qquad V_{\text{Celine}} = \{b,\ c\} \tag{2.26}$$

4. The set of possible outcomes O consists of the four terminal leaf payoff pairs:

$$O = \{(3,\ 2),\ (1,\ 1),\ (0,\ 0),\ (2,\ 3)\} \tag{2.27}$$

where each pair gives (Bob's utility, Celine's utility).

5. The mapping u from each terminal node to an element of O is:

- Sports–Sports leaf $\mapsto (2,\ 3)$
- Sports–Comedy leaf $\mapsto (0,\ 0)$
- Comedy–Sports leaf $\mapsto (1,\ 1)$
- Comedy–Comedy leaf $\mapsto (3,\ 2)$

Solution 2.2: Solution to Exercise 2

We refer to the Figure 2.5 in which neither player observes the other's choice.

1. The finite set of players is:

$$\mathcal{N} = \{\text{Bob},\ \text{Celine}\} \tag{2.28}$$

so $|\mathcal{N}| = 2$.

2. The game tree $G = (V, E, x^0)$ consists of:

- **Vertices** V: the root node a (Bob's decision), two intermediate nodes b and c (which belong to the same information set for Celine), and four terminal leaf nodes.
- **Edges** E: the edge from a to b (Comedy), the edge from a to c (Sports), and four edges from b and c to the terminal nodes (each labelled Sports or Comedy).
- **Root** $x^0 = a$.

3. The partition $(V_i)_{i \in \mathcal{N}}$ is:

$$V_{\text{Bob}} = \{a\}, \qquad V_{\text{Celine}} = \{b,\ c\} \tag{2.29}$$

4. The set of possible outcomes O consists of the four terminal leaf payoff pairs:

$$O = \{(3,\ 2),\ (1,\ 1),\ (0,\ 0),\ (2,\ 3)\} \tag{2.30}$$

5. The mapping u from each terminal node to an element of O is the same as in the perfect information game:

- Sports–Sports leaf $\mapsto (2,\ 3)$
- Sports–Comedy leaf $\mapsto (0,\ 0)$
- Comedy–Sports leaf $\mapsto (1,\ 1)$
- Comedy–Comedy leaf $\mapsto (3,\ 2)$

The key structural difference from the perfect information game is that nodes b and c belong to the **same information set** for Celine, represented by a dashed line connecting them, because Celine cannot observe Bob's choice before making her own.

Solution 2.3: Solution to Exercise 3

For each game with $\mathcal{N} = \{\text{Alice},\ \text{Bob}\}$, decision nodes labelled A_i belong to Alice and B_i belong to Bob. Information sets partition each player's decision nodes into groups that player cannot distinguish.

1. **Game 1 (perfect information, Alice moves first):**

The game tree has Alice at the root with a single decision node. Bob has two decision nodes reached after each of Alice's choices, and he can observe Alice's move. Since all information is available at each node, every node forms its own singleton information set:

$$v_{\text{Alice}} = \{\ \{A\}\ \} \qquad v_{\text{Bob}} = \{\ \{B_1\},\ \{B_2\}\ \} \tag{2.31}$$

2. **Game 2 (imperfect information):**

Alice moves first. Bob has two decision nodes but cannot distinguish between them (they are connected by a dashed line). Thus Bob's two nodes form a single information set:

$$v_{\text{Alice}} = \{\ \{A\}\ \} \qquad v_{\text{Bob}} = \{\ \{B_1,\ B_2\}\ \} \tag{2.32}$$

3. **Game 3 (perfect information, multiple rounds):**

All players observe all previous moves; each node is a singleton information set:

$$v_{\text{Alice}} = \{\ \{A_1\},\ \{A_2\}\ \} \qquad v_{\text{Bob}} = \{\ \{B_1\}, \{B_2\}\ \} \tag{2.33}$$

(or whatever the specific structure of the tree implies, with each node forming its own information set since information is perfect).

4. **Game 4 (imperfect information, multiple rounds):**

Alice has a single decision node at the root. Bob has multiple decision nodes that are grouped into information sets according to which of Alice's moves he can observe. If Bob cannot distinguish between two of his nodes (e.g. B_1 and B_2 are connected by a dashed line), they form one information set:

$$v_{\text{Alice}} = \{\ \{A_1\}, \{A_2\}\ \} \qquad v_{\text{Bob}} = \{\ \{B_1,\ B_2\}\ \} \tag{2.34}$$

5. **Game 5 (incoherent imperfect information):**

This game is **incoherent** because nodes grouped into the same information set do not have the same number of successors (or the successors have different action labels). This violates the requirement that all vertices in an information set must have identical available actions. Therefore, the depicted grouping is **not a valid information set structure**: a well-formed extensive form game cannot have an information set where the nodes have different action sets available.

Solution 2.4: Solution to Exercise 4

We model the three-player messaging platform coordination game.

Players and strategy sets:

The players are:

$$\mathcal{N} = \{\text{Alice},\ \text{Bob},\ \text{Celine}\} \tag{2.35}$$

Each player has the same action set:

$$\mathcal{A}_i = \{\text{WhatsApp},\ \text{Instagram},\ \text{Snapchat}\} \qquad \text{for } i \in \mathcal{N} \tag{2.36}$$

Interpretation of utility values:

A reasonable interpretation is that each player's utility equals the number of the other two friends who choose the same platform. The motivation is that each person gains value from being able to communicate with friends: more shared connections means higher utility.

Under this interpretation:

- Utility $= 2$ if all three choose the same platform.
- Utility $= 1$ if exactly two of the three choose the same platform (and the player is one of the two matching).
- Utility $= 0$ if the player is alone on their chosen platform.

Normal form representation:

Since there are three players, the game cannot be expressed as a single bi-matrix. Instead, for each choice of Celine's action, we write a bi-matrix for Alice (row) and Bob (column):

Using W = WhatsApp, I = Instagram, S = Snapchat, and payoff triples (Alice, Bob, Celine):

Celine plays W:

$$\begin{array}{c|ccc} & W & I & S \\ W & (2,2,2) & (1,0,1) & (1,0,1) \\ I & (0,1,1) & (1,1,0) & (0,0,0) \\ S & (0,1,1) & (0,0,0) & (1,1,0) \end{array} \tag{2.37}$$

Celine plays I:

$$\begin{array}{c|ccc} & W & I & S \\ W & (1,1,0) & (0,1,1) & (0,0,0) \\ I & (1,0,1) & (2,2,2) & (1,0,1) \\ S & (0,0,0) & (0,1,1) & (1,1,0) \end{array} \tag{2.38}$$

Celine plays S:

$$\begin{array}{c|ccc} & W & I & S \\ W & (1,1,0) & (0,0,0) & (0,1,1) \\ I & (0,0,0) & (1,1,0) & (0,1,1) \\ S & (1,0,1) & (1,0,1) & (2,2,2) \end{array} \tag{2.39}$$

Below is code that creates this game using Gambit:

```
import itertools
import pygambit as gbt

g = gbt.Game.new_table([3, 3, 3])
p1, p2, p3 = g.players

for i, j, k in itertools.product(range(3), repeat=3):
    same_as_p1 = (j == i) + (k == i)
    same_as_p2 = (i == j) + (k == j)
    same_as_p3 = (i == k) + (j == k)
    g[(i, j, k)][p1] = same_as_p1
    g[(i, j, k)][p2] = same_as_p2
    g[(i, j, k)][p3] = same_as_p3

print(g)
```

```
Game(title='Untitled strategic game')
```

Solution 2.5: Solution to Exercise 5

Players and strategy sets:

The players are the two countries, which we call Country 1 (row player) and Country 2 (column player). Each has the action set:

$$\mathcal{A}_i = \{\text{Peace},\ \text{Attack}\} \tag{2.40}$$

Payoff matrices:

Using utility equal to the negative of casualties, the payoff matrices are:

$$M_r = \begin{pmatrix} 0 & -15000 \\ -13000 & -10000 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0 & -13000 \\ -15000 & -10000 \end{pmatrix} \tag{2.41}$$

where rows correspond to Country 1's actions (Peace, Attack) and columns to Country 2's actions (Peace, Attack).

Expected utility plots:

Let $\sigma_1 = (x, 1-x)$ be Country 1's strategy (probability x of playing Peace) and $\sigma_2 = (y, 1-y)$ be Country 2's strategy (probability y of playing Peace).

When Country 2 plays Peace ($y = 1$, $\sigma_2 = (1, 0)$):

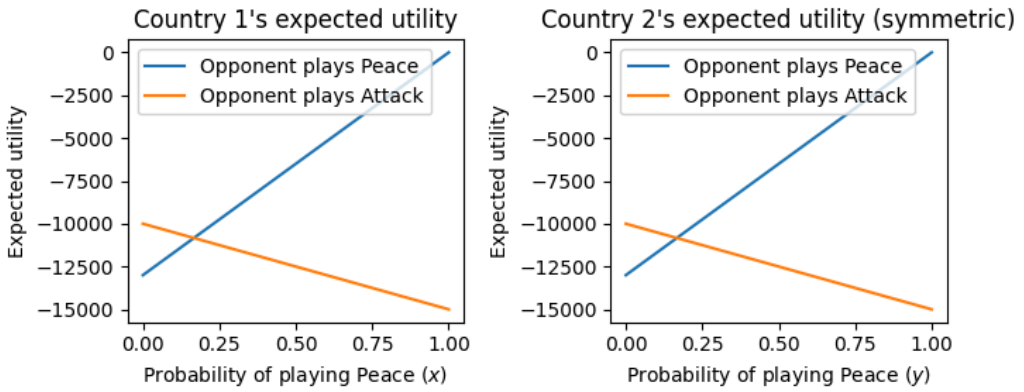
$$u_1(\sigma_1, \text{Peace}) = x \cdot 0 + (1-x) \cdot (-13000) = 13000x - 13000 \tag{2.42}$$

When Country 2 plays Attack ($y = 0$, $\sigma_2 = (0, 1)$):

$$\begin{aligned} u_1(\sigma_1, \text{Attack}) &= x \cdot (-15000) + (1-x) \cdot (-10000) \\ &= -15000x - 10000(1-x) \\ &= -5000x - 10000 \end{aligned} \tag{2.43}$$

By symmetry the same holds for Country 2 with the roles swapped.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 1, 100)

u1_given_peace = -13000 * (1 - x)
u1_given_attack = -15000 * x - 10000 * (1 - x)

plt.figure(figsize=(7, 3))

plt.subplot(1, 2, 1)
plt.plot(x, u1_given_peace, label="Opponent plays Peace")
plt.plot(x, u1_given_attack, label="Opponent plays Attack")
plt.xlabel("Probability of playing Peace ($x$)")
plt.ylabel("Expected utility")
plt.title("Country 1's expected utility")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(x, u1_given_peace, label="Opponent plays Peace")
plt.plot(x, u1_given_attack, label="Opponent plays Attack")
plt.xlabel("Probability of playing Peace ($y$)")
plt.ylabel("Expected utility")
plt.title("Country 2's expected utility (symmetric)")
plt.legend()

plt.tight_layout()
```



3. Rationality

A rational player asks what is the best response to the strategies of others, and assumes that others reason the same way. This chapter formalises that idea, developing dominance, best responses, and iterated elimination to predict what rational players will and will not do.

3.1 Motivating Example: Competing Restaurant Locations

Two restaurant chains, **FreshBite** and **UrbanEats**, are expanding into a new city. Each must choose one of three neighbourhoods for a flagship store: **Downtown**, **Midtown**, or **Suburb**.

- **FreshBite** prioritises foot traffic and brand visibility.
- **UrbanEats** prefers low rental costs and long-term margins.

The projected profits (in £1000s) for each company depend on both their own choice and their competitor's. Since they are competing in the same market, outcomes reflect strategic interaction.

Table 3.1 and Table 3.2 show the expected profits for each company based on their own decision (row) and their competitor's choice (column).

	Downtown	Midtown	Suburb
Downtown	4	3	2
Midtown	3	2	3
Suburb	2	1	1

Table 3.1: Projected profits for **FreshBite** (in thousands of pounds)

	Downtown	Midtown	Suburb
Downtown	1	0	2
Midtown	0	2	6
Suburb	2	4	5

Table 3.2: Projected profits for **UrbanEats** (in thousands of pounds)

These tables are assumed to be publicly available from a consultation commissioned by the city council.

> **Important**
>
> Table 3.1 and Table 3.2 are not the payoff matrices M_r and M_c of (2.2). They represent the raw projected profit data used to define the game.

The payoff matrices for the equivalent Normal Form Game are:

$$M_r = \begin{pmatrix} 4 & 3 & 2 \\ 3 & 2 & 3 \\ 2 & 1 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 2 & 4 \\ 2 & 6 & 5 \end{pmatrix} \tag{3.1}$$

> **Note**
>
> M_c is obtained by transposing the rows and columns of Table 3.2.

Let us begin from FreshBite's perspective: Suburb results in lower profits than Midtown or Downtown, regardless of UrbanEats' action. A rational FreshBite would therefore never choose Suburb, as illustrated in Figure 3.1 although it might still choose **Midtown** or **Downtown**.

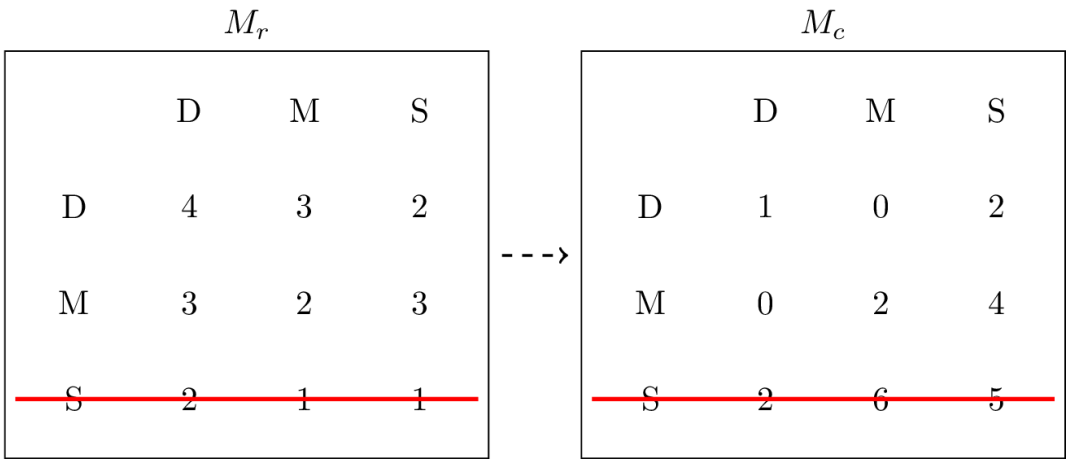


Figure 3.1: Crossing out Suburb as an action that will never be played by the row player.

With Suburb eliminated for FreshBite, UrbanEats can update their view. Against the remaining options (Downtown and Midtown), UrbanEats consistently receives higher profits by choosing Suburb rather than either alternative. Hence, Downtown and Midtown can be removed from consideration, as shown in Figure 3.2.

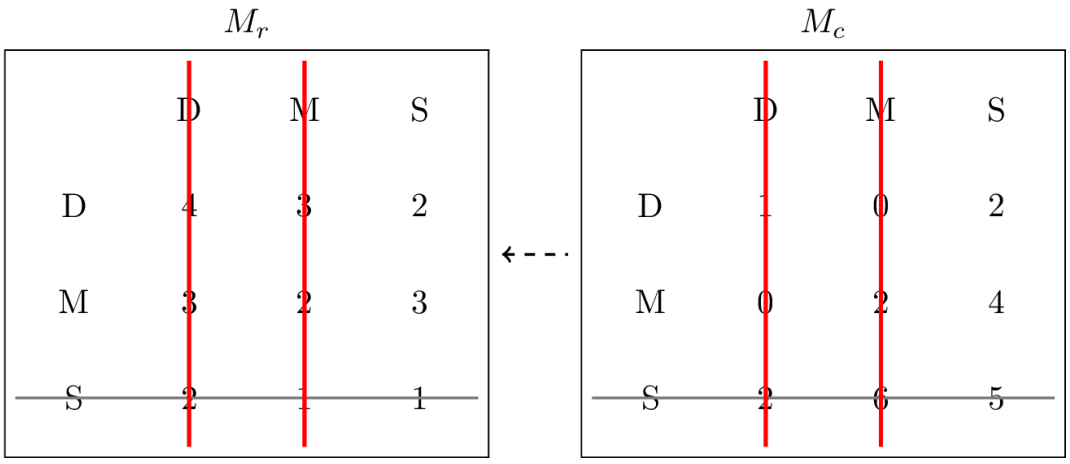


Figure 3.2: Crossing out Midtown and Downtown as actions that will never be played by the column player.

FreshBite, now comparing Downtown and Midtown against only Suburb from UrbanEats, finds that Midtown yields a better outcome. They therefore eliminate Downtown, leaving Midtown as their only rational option, as shown in Figure 3.3.

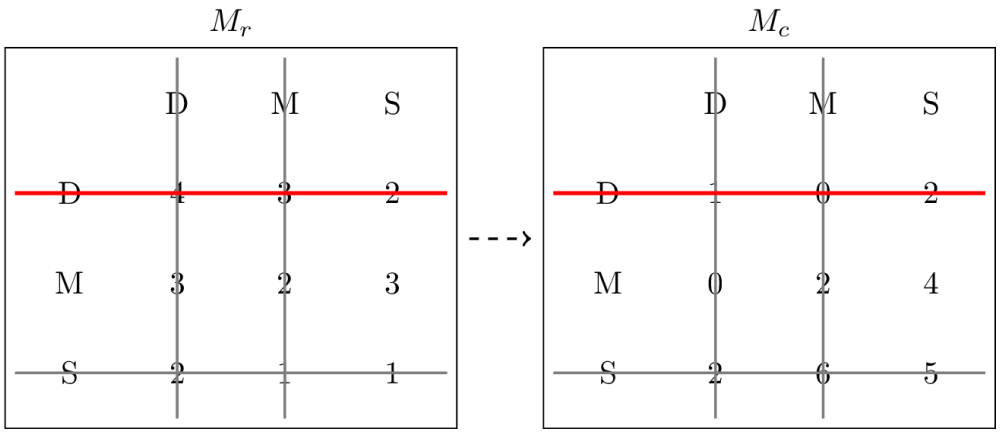


Figure 3.3: Crossing out Downtown as an action that will never be played by the row player.

The step-by-step logic of **iterated elimination of strategies**, combined with the assumption of full information and mutual rationality, leads to the predicted outcome:

- FreshBite opens in **Midtown**: with an expected profit of £3,000.
- UrbanEats opens in **Suburb**: with an expected profit of £4,000

> **Note**
>
> This outcome is not necessarily optimal for both players, but it is rational given the available information and the structure of the interaction.

In the next section, we introduce the **theory of dominance and best responses**. These concepts provide the formal tools for identifying outcomes like this one, and for understanding how strategic agents simplify complex decisions.

3.2 Theory

In certain games it is evident that certain strategies should never be used by a rational player. To formalise this we need a couple of definitions.

3.2.1 Definition: Incomplete Strategy Profile

In an N-player normal form game, when focusing on player i, we denote by $s_{(-i)}$ an **incomplete strategy profile** representing the strategies chosen by all players *except* player i.

For example, in a 3-player game where $\mathcal{A}_i = \{A, B\}$ for all i, a valid strategy profile might be $s = ((1,0),\ (1,0),\ (0,1))$, indicating that players 1 and 2 choose A, and player 3 chooses B.

The corresponding incomplete strategy profile for player 2 is: $s_{(-2)} = ((1,0),\ (0,1))$, capturing only the choices of players 1 and 3.

We write $S_{(-i)}$ to denote the set of all such incomplete strategy profiles from the perspective of player i.

This notation allows us to define a key concept in strategic reasoning.

3.2.2 Definition: Strictly Dominated Strategy

In an N-player normal form game, an action $a_i \in \mathcal{A}_i$ is said to be **strictly dominated** if there exists a strategy $\sigma_i \in \Delta(\mathcal{A}_i)$ such that:

$$u_i\big(\sigma_i, s_{(-i)}\big) > u_i\big(a_i, s_{(-i)}\big) \quad \text{for all } s_{(-i)} \in S_{(-i)}. \tag{3.2}$$

That is, no matter what the other players do, player i always receives a strictly higher payoff by playing σ_i instead of the action a_i.

When modelling rational behaviour, such actions can be systematically excluded from consideration.

3.2.3 Definition: Weakly Dominated Strategy

In an N-player normal form game, an action $a_i \in \mathcal{A}_i$ is said to be **weakly dominated** if there exists a strategy $\sigma_i \in \Delta(\mathcal{A}_i)$ such that:

$$u_i(\sigma_i, s_{-i}) \geq u_i(a_i, s_{-i}) \quad \text{for all } s_{-i} \in S_{-i}, \tag{3.3}$$

and

$$u_i(\sigma_i, \bar{s}) > u_i(a_i, \bar{s}) \quad \text{for some } \bar{s} \in S_{-i}. \tag{3.4}$$

That is, σ_i does at least as well as s_i against all possible strategies of the other players, and strictly better against at least one.

As with strictly dominated strategies, we can use weak dominance to guide predictions of rational behaviour by eliminating weakly dominated strategies. Doing so, however, relies on a deeper assumption about what players know about one another.

3.2.4 Common Knowledge of Rationality

So far, we have assumed that players behave rationally. To justify iterative elimination procedures, we must go further. The key idea is that not only are players rational, but they also know that others are rational, and that this knowledge is shared recursively.

This leads to the concept of **Common Knowledge of Rationality (CKR)**:

- Each player is rational.
- Each player knows that all other players are rational.
- Each player knows that all others know that they are rational.
- Each player knows that all others know that they all know that they are rational.
- And so on, indefinitely.

This infinite chain of mutual knowledge is what constitutes **common knowledge**.

By assuming CKR, we justify the use of techniques like the iterated elimination of dominated strategies as a model of rational prediction.

3.2.4.1 Example: Predicted Behaviour through Iterated Elimination Strategies

Consider the following normal form game, with separate payoff matrices for the row and column players:

$$M_r = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0 & 2 & 1 \\ 3 & 1 & 1 \end{pmatrix} \tag{3.5}$$

We proceed by examining whether any strategies can be removed based on weak dominance.

Step 1: From the column player's perspective, column c_3 is weakly dominated by column c_2. That is, for every row, c_2 performs at least as well as c_3, and strictly better for at least one row. We remove c_3 from consideration.

Step 2: With c_3 removed, the row player now compares r_1 and r_2. Row r_2 is weakly dominated by r_1, since r_1 yields outcomes that are always at least as good and sometimes strictly better. We eliminate r_2.

Step 3: Finally, the column player is left with c_1 and c_2. Column c_1 is now dominated by c_2, and can be eliminated.

After these steps, we are left with a single strategy profile:

$$s = (r_1, c_2) \tag{3.6}$$

This strategy profile is a **predicted outcome** based on iterated elimination of weakly dominated strategies, assuming mutual rationality and full knowledge of the payoffs.

Note

Unlike strictly dominated strategies, weakly dominated strategies can sometimes be part of a predicted outcome. This process provides a reasonable prediction but not a definitive one.

3.2.4.2 Example: eliminating strategies does not suffice

In many cases, we can simplify a game by eliminating dominated strategies. However, not all games can be fully resolved using this approach.

Consider the following game.

$$M_r = \begin{pmatrix} 1 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 3 & 5 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 2 & 2 \\ 0 & 3 & 1 \\ 5 & 4 & 6 \end{pmatrix} \tag{3.7}$$

Although some weakly dominated strategies may be eliminated here, we still end up with multiple strategy combinations remaining. No single obvious solution emerges from dominance alone. Here is another example:

$$M_r = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix} \tag{3.8}$$

In this game, no strategy is strictly or weakly dominated for either player. Elimination techniques do not reduce the strategy space, and further analysis is required to predict behaviour.

These examples show that **dominance is a helpful tool**, but not always sufficient. We need further concepts, such as **best responses** to analyse such games. Payoff matrices for the row and column players:

3.2.5 Definition: Best Response

In an N-player normal form game, a strategy s^* for player i is a **best response** to some incomplete strategy profile s_{-i} if and only if:

$$u_i(s^*,\ s_{-i}) \geq u_i(s,\ s_{-i}) \quad \text{for all } s \in \Delta(\mathcal{A}_i). \tag{3.9}$$

That is, s^* gives player i the highest possible payoff, given the strategies chosen by the other players.

We can now begin to predict rational behaviour by identifying the best responses to actions of the other players.

3.2.5.1 Example: Predicted Behaviour through Best Responses in the Action Space

Consider the following game with payoff matrices for the row and column players:

$$M_r = \begin{pmatrix} 1 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 3 & 5 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 2 & 2 \\ 0 & 3 & 1 \\ 5 & 4 & 6 \end{pmatrix} \tag{3.10}$$

We highlight best responses by underlining the maximum value(s) in each row (for the column player) and in each column (for the row player):

$$M_r = \begin{pmatrix} 1 & \underline{4} & 2 \\ \underline{4} & 0 & 4 \\ 2 & 3 & \underline{5} \end{pmatrix} \qquad M_c = \begin{pmatrix} \underline{3} & 2 & 2 \\ 0 & \underline{3} & 1 \\ 5 & 4 & \underline{6} \end{pmatrix} \tag{3.11}$$

Here, $((0,0,1),(0,0,1))$ is a pair of best responses:

- r_3 is a best response to c_3,
- and c_3 is a best response to r_3.

This mutual best response suggests a potential **stable outcome**.

In the next example we will consider finding best responses in the entire strategy space and not just the action space.

3.2.5.2 Example: Predicted Behaviour through Best Responses in the Strategy Space

We can also identify best responses when players use **strategies**. Let us begin by examining the **Matching Pennies** game.

3.2.5.2.1 Example: Matching Pennies

Recalling the Payoff matrices for the row and column players:

$$M_r = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.12}$$

Assume that player 2 plays a strategy $\sigma_2 = (x,\ 1-x)$. Then player 1's expected utilities are:

$$u_1(r_1,\ \sigma_2) = 2x - 1 \qquad \text{and} \qquad u_1(r_2,\ \sigma_2) = 1 - 2x \tag{3.13}$$

These two utilities are shown in Figure 3.4.

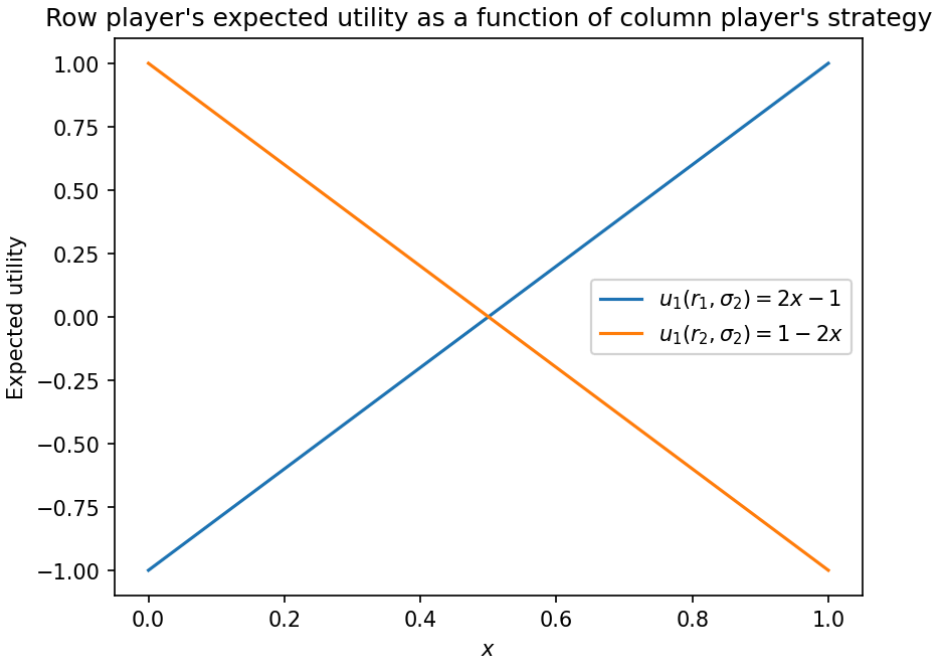


Figure 3.4: Row player's expected utility for each action as a function of the probability x that the column player plays c_1 in Matching Pennies.

From the graph we observe:

1. If $x < \frac{1}{2}$, then r_2 is a best response for player 1.
2. If $x > \frac{1}{2}$, then r_1 is a best response for player 1.
3. If $x = \frac{1}{2}$, then player 1 is indifferent between r_1 and r_2.

3.2.5.3 Examples: Coordination Game

Recalling the Payoff matrices for the row and column players:

$$M_r = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix} \tag{3.14}$$

Assume that player 1 plays a strategy $\sigma_1 = (x,\ 1-x)$. Then player 2's expected utilities are:

$$u_2(c_1,\ \sigma_1) = 2x + 0(1-x) = 2x \qquad \text{and} \qquad u_2(c_2,\ \sigma_1) = 1x + 3(1-x) = 3 - 2x \tag{3.15}$$

These two utilities are shown in Figure 3.5.

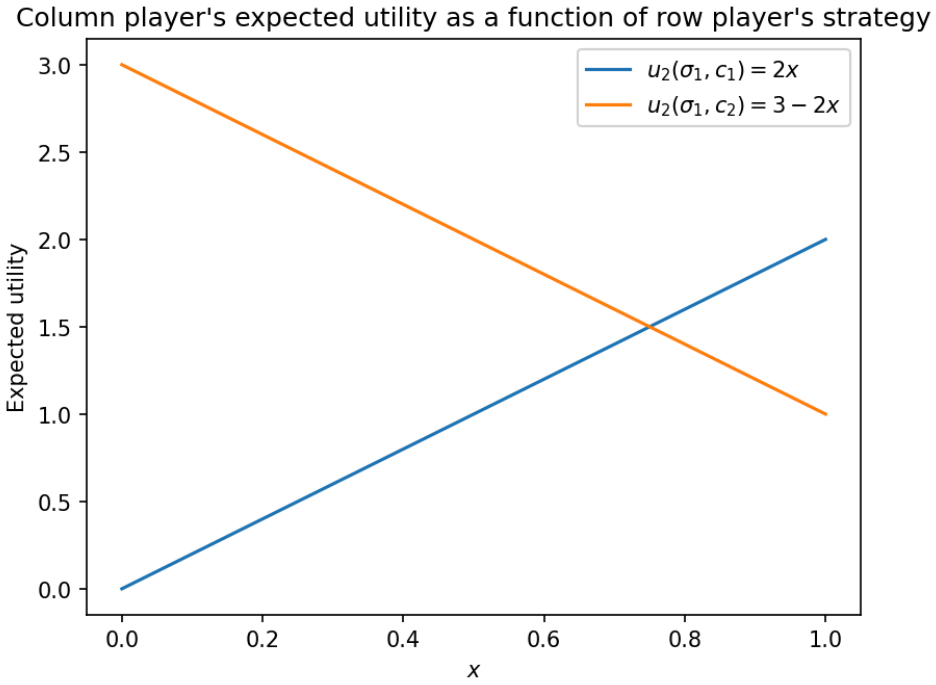


Figure 3.5: Column player's expected utility for each action as a function of the probability x that the row player plays r_1 in the Coordination Game.

We can now determine the best response for player 2 based on the value of x:

1. If $x > \frac{3}{4}$, then c_1 is a best response.
2. If $x < \frac{3}{4}$, then c_2 is a best response.
3. If $x = \frac{3}{4}$, then player 2 is **indifferent** between c_1 and c_2.

This example shows how best responses depend continuously on the opponent's strategy, and that indifference can arise at precise thresholds in strategies.

3.2.6 Theorem: Best Response Condition

In a two-player game $(A,\ B) \in (\mathbb{R}^{m \times n})^2$, a strategy σ_r^* of the row player is a **best response** to a strategy σ_c of the column player if and only if:

$$\sigma_{r^*}(i) > 0 \quad \Rightarrow \quad \left(A\sigma_c^T\right)_i = \max_{k \in \mathcal{A}_1} \left(A\sigma_c^T\right)_k \quad \text{for all } i \in \mathcal{A}_1 \tag{3.16}$$

3.2.6.1 Proof

The term $\left(A\sigma_c^T\right)_i$ represents the utility for the row player when playing their i^{th} action. Thus:

$$\sigma_r A\sigma_c^T = \sum_{i=1}^{m} \sigma_r(i) \cdot \left(A\sigma_c^T\right)_i \tag{3.17}$$

Let $u = \max_k \left(A\sigma_c^T\right)_k$. Then:

$$\begin{aligned}\sigma_r A\sigma_c^T &= \sum_{i=1}^{m} \sigma_r(i)\left[u - u + \left(A\sigma_c^T\right)_i\right] \\ &= \sum_{i=1}^{m} \sigma_r(i)u - \sum_{i=1}^{m} \sigma_r(i)\left(u - \left(A\sigma_c^T\right)_i\right) \\ &= u - \sum_{i=1}^{m} \sigma_r(i)\left(u - \left(A\sigma_c^T\right)_i\right)\end{aligned} \tag{3.18}$$

Since $u - \left(A\sigma_c^T\right)_i \geq 0$ for all i, the maximum expected utility for the row player is u, and this occurs **if and only if**:

$$\sigma_r(i) > 0 \quad \Rightarrow \quad \left(A\sigma_c^T\right)_i = u \tag{3.19}$$

as required.

3.2.6.2 Example: Rock Paper Scissors Lizard Spock

The classic Rock Paper Scissors game can be extended to **Rock Paper Scissors Lizard Spock**, where:

- Scissors cuts Paper
- Paper covers Rock
- Rock crushes Lizard
- Lizard poisons Spock
- Spock smashes Scissors
- Scissors decapitates Lizard
- Lizard eats Paper
- Paper disproves Spock
- Spock vaporises Rock
- Rock crushes Scissors

This results in the following payoff matrices:

$$M_r = \begin{pmatrix} 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & -1 & -1 & 1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 0 \end{pmatrix} \qquad M_c = -M_r \tag{3.20}$$

> **Note**
>
> This is an example of a Zero-Sum Game, where one player's gain is exactly the other's loss.

We can now use the Best Response Condition Theorem to check whether the strategy $\sigma_1 = \left(\frac{1}{3},\ 0,\ \frac{1}{3},\ \frac{1}{3},\ 0\right)$ is a best response to $\sigma_2 = \left(\frac{1}{4},\ 0,\ 0,\ \frac{3}{4},\ 0\right)$.

We begin by computing the expected utilities for each of player 1's actions:

$$M_r\sigma_2^T = \begin{pmatrix} 0.75 \\ -0.5 \\ 0.5 \\ -0.25 \\ -0.5 \end{pmatrix} \tag{3.21}$$

From this vector, we observe that:

$$\max_{k \in \mathcal{A}_1} \left(A\sigma_2^T\right)_k = 0.75 \tag{3.22}$$

Note

The set of actions that a strategy σ plays with non zero probability is referred to as the support of σ.

The support of σ_1 is $\mathcal{S}(\sigma_1) = \{1,\ 3,\ 4\}$. Among these, only $\left(A\sigma_2^T\right)_1 = 0.75$ equals the maximum.

Since $\left(A\sigma_2^T\right)_3 = 0.5$ and $\left(A\sigma_2^T\right)_4 = -0.25$, the condition for a best response is not satisfied.

Therefore, σ_1 is not a best response to σ_2. Indeed, the row player should revise their strategy to place positive weight only on the action that yields the maximum payoff.

3.3 Exercises

Exercise 3.1:

Use **iterated elimination of dominated strategies** to attempt to predict rational behaviour in each of the following games. Represent all steps clearly.

1.

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix} \tag{3.23}$$

1.

$$A = \begin{pmatrix} 2 & 1 & 3 & 17 \\ 27 & 3 & 1 & 1 \\ 4 & 6 & 7 & 18 \end{pmatrix} \qquad B = \begin{pmatrix} 11 & 9 & 10 & 22 \\ 0 & 1 & 1 & 0 \\ 2 & 10 & 12 & 0 \end{pmatrix} \tag{3.24}$$

1.

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} \tag{3.25}$$

1.

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{3.26}$$

Exercise 3.2:

For each game in Exercise 1:

1. Identify **all best response** actions.
2. Attempt to predict **rational outcomes** using best response reasoning.
3. Explain any instances where this approach **fails to fully determine the outcome** (e.g. games with multiple best responses).

Exercise 3.3:

Model the following real-world conflict scenario as a **normal form game**.

Two neighbouring countries possess powerful military forces.

- If both attack, each suffers 10,000 civilian casualties.
- If one attacks while the other remains peaceful, the peaceful country suffers 15,000 casualties and retaliates, causing the attacker 13,000 casualties.
- If both remain peaceful, there are no casualties.

Answer the following:

1. Clearly state the **players** and their **action sets**.
2. Represent the game in **normal form**, assuming utilities are the negative of casualties.
3. Plot the **expected utilities** for each country assuming the other country plays a generic strategy while it plays a strategy that always chooses one of the actions:
    - Peaceful
    - Aggressive
4. Identify the **best responses** of each country.

Exercise 3.4:

Two shops on the same street are deciding how to price their product. Each can set the price as **Low**, **Medium**, or **High**.

- If both choose the same price, they split the market.
- If one sets a lower price, they attract more customers but earn less per item.
- If one sets a higher price, they retain fewer customers but may benefit from higher margins.

1. Define a plausible utility table for both players.
2. Represent the game in normal form using two matrices.
3. Identify and eliminate any actions that are never chosen under rational behaviour.
4. Discuss whether this game has any pairs of actions that are best responses to each other.

Exercise 3.5:

Two workers must each choose a task to complete in a shared workspace: **Documentation**, **Coding**, or **Debugging**.

- If they choose the same task, productivity drops due to coordination overhead.
- If they choose different tasks, overall productivity improves.
- Each task has a different utility value depending on the player's preference and skill.

1. Construct a utility matrix for both players that reflects this scenario.
2. Represent the game in normal form.
3. Are any actions strictly or weakly dominated?
4. Determine best responses for both players.

3.4 Programming

3.4.1 Computing utilities of all actions

Theorem: Best Response Condition relies on computing the utilities of all actions when the other player is playing a specific strategy. This corresponds to the matrix multiplication:

$$M_r \sigma_c^T \tag{3.27}$$

Similarly to Linear Algebraic Form of Expected Utility this enables **efficient numerical computation** in programming languages that support vectorized matrix operations.

3.4.2 Computing utilities of all actions with Numpy

Python's numerical library **NumPy** (Harris et al., 2020) provides vectorized operations through its `array` class. Below, we define the elements from the earlier Rock Paper Scissors Lizard Spock example:

```
import numpy as np

M_r = np.array(
    (
        (0, -1, 1, 1, -1),
        (1, 0, -1, -1, 1),
        (-1, 1, 0, 1, -1),
        (-1, 1, -1, 0, 1),
        (1, -1, 1, -1, 0),
    )
)

sigma_2 = np.array((1 / 4, 0, 0, 3 / 4, 0))
```

We can now compute the expected utility for each of the row player's actions:

```
print(f"Expected utility of each action: {M_r @ sigma_2}")
```

```
Expected utility of each action: [ 0.75 -0.5   0.5  -0.25 -0.5 ]
```

3.4.3 Finding non zero entries of an array with Numpy

NumPy gives functionality for finding non zero entries of an array:

```
sigma_1 = np.array((1 / 3, 0, 1 / 3, 1 / 3, 0))
print(f"Indices of actions played: {sigma_1.nonzero()}")
```

```
Indices of actions played: (array([0, 2, 3]),)
```

3.4.4 Checking the best response condition with Numpy

To check the best response condition with NumPy we can efficiently compare the location of the non zero values of a strategy with the maximum utility of the action set.

```
played_actions_maximise = (M_r @ sigma_2)[sigma_1.nonzero()] == (M_r @
sigma_2).max()
print(f"Each played action achieves the maximum utility:
{played_actions_maximise}")
```

```
Each played action achieves the maximum utility: [ True False False]
```

Just as in Example: Rock Paper Scissors Lizard Spock we see that σ_1 is not a best response as all the non zero values do not give maximum utility. If we modify the row player's behaviour to only play the first action then we do get a best response:

```
sigma_1 = np.array((1, 0, 0, 0, 0))
played_actions_maximise = (M_r @ sigma_2)[sigma_1.nonzero()] == (M_r @
sigma_2).max()
print(f"Each played action achieves the maximum utility:
{played_actions_maximise}")
```

```
Each played action achieves the maximum utility: [ True]
```

3.4.5 Checking best response condition with Nashpy

The Python library **Nashpy** (Knight & Campbell, 2018) can be used to directly check the best response condition for two strategies. It checks if either strategy is a best response to each other.

```
import nashpy as nash

rpsls = nash.Game(M_r, - M_r)
print(f"Best response check (sigma_1, sigma_2): "
      f"{rpsls.is_best_response(sigma_r=sigma_1, sigma_c=sigma_2)}")
```

```
Best response check (sigma_1, sigma_2): (True, False)
```

This confirms that $\sigma_1 = (1, 0, 0, 0, 0)$ is a best response to $\sigma_2 = (1/4, 0, 0, 3/4, 0)$ but not vice versa.

3.5 Notable Research

In (Nash Jr, 1950), John Nash introduced the foundational concept of **Nash equilibrium**: a strategy profile (i.e., a combination of strategies, one per player) where each strategy is a best response to the others. In such a setting, no player has an incentive to unilaterally deviate from their choice. Nash's landmark result showed that at least one such equilibrium exists in any finite game, a result that earned him the Nobel Prize in Economics.

More recent research explores how often certain types of equilibria occur. For instance, (Wiese & Heinrich, 2022) investigates how frequently **best response dynamics** (where players iteratively switch to their best response) lead to convergence at a Nash equilibrium. This builds on earlier work such as (Dresher, 1970), which studied the likelihood that a game possesses a Nash equilibrium in actions.

Another line of research explores how the structure of a game influences the convergence to equilibrium. For example, in congestion games or routing games, players choose paths through a network, and each player's cost depends on the congestion caused by others. These games often admit Nash equilibria in actions and allow best response dynamics to converge under mild conditions. This property has led to applications in traffic flow, internet routing, and load balancing. The work of

(Rosenthal, 1973) formally introduced this class of games and remains a cornerstone of algorithmic game theory.

In an applied context, (Knight et al., 2017) models interactions between hospitals under performance-based targets. The authors prove the existence of an equilibrium in **actions** (as opposed to more general strategies). This finding provides insight into how policy incentives might be structured to align hospital decision-making with public health outcomes.

3.6 Conclusion

In this chapter, we introduced fundamental tools for predicting strategic behaviour: **dominated strategies**, **best responses**, and the assumption of **common knowledge of rationality**. These ideas allowed us to identify outcomes that rational players would be unlikely to avoid, such as the elimination of actions that never perform best and the selection of strategies that respond optimally to others.

Iterated elimination and **best response analysis** reveal plausible behaviours in one-shot games, and both extend naturally to **strategies** where probability plays a role.

These concepts (summarised in Table 3.3) lay the foundation for understanding **equilibrium**, the central concept where players' strategies mutually reinforce each other. In the next chapter, we formalise this idea, introducing **Nash equilibrium** and exploring its existence, interpretation, and implications across different types of games.

Concept	Description
Dominated Strategy	An action that is always worse than another; never rational to play
Iterated Elimination	Sequentially removing dominated strategies to simplify the game
Best Response	The strategy that maximises a player's utility given others' choices
Strategy Profile	A complete specification of strategies for all players
Incomplete Strategy Profile	Strategies for all players except one
Common Knowledge of Rationality	All players are rational, and know that others are, recursively
Best Response Condition	A formal test for optimality in response to a strategy

Table 3.3: Summary of core concepts introduced in this chapter.

Attention

If a strategy is never a best response to any opponent strategy, it can be ignored when predicting rational behaviour.

3.7 Solutions

Solution 3.1: Solution to Exercise 1

We apply iterated elimination of dominated strategies to each game.

Game 1:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix} \tag{3.28}$$

Step 1 (Column player): Compare $c_1 = (1, 1)$ and $c_2 = (1, 3)$. We see that c_2 weakly dominates c_1.

Step 2 (Row player): With only c_2 remaining, both r_1 and r_2 give the row player a payoff of 1 against c_2. Neither row is dominated, so the elimination process terminates here.

Predicted outcome: The process eliminates c_1, predicting that the column player plays c_2. Both r_1 and r_2 are consistent with iterated elimination.

Game 2:

$$A = \begin{pmatrix} 2 & 1 & 3 & 17 \\ 27 & 3 & 1 & 1 \\ 4 & 6 & 7 & 18 \end{pmatrix} \qquad B = \begin{pmatrix} 11 & 9 & 10 & 22 \\ 0 & 1 & 1 & 0 \\ 2 & 10 & 12 & 0 \end{pmatrix} \tag{3.29}$$

Step 1 (Row player): Compare each row:

- $r_1 = (2, 1, 3, 17)$
- $r_2 = (27, 3, 1, 1)$
- $r_3 = (4, 6, 7, 18)$

Comparing r_1 and r_3: r_3 gives $4 > 2, 6 > 1, 7 > 3, 18 > 17$ against all columns. So r_1 is **strictly dominated** by r_3. Eliminate r_1.

Step 2 (Column player): With r_2 and r_3 remaining, the column payoffs are:

$$B' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 2 & 10 & 12 & 0 \end{pmatrix} \tag{3.30}$$

Compare each column for the column player. Looking at c_4: against r_2 it gives 0, against r_3 it gives 0. Column c_1: against r_2 gives 0, against r_3 gives 2. So c_4 is weakly dominated by c_1 (same against r_2, worse against r_3). Eliminate c_4.

Step 3 (Column player): With columns c_1, c_2, c_3 remaining:

$$B'' = \begin{pmatrix} 0 & 1 & 1 \\ 2 & 10 & 12 \end{pmatrix} \tag{3.31}$$

Compare c_1 vs c_2: c_2 gives $1 > 0$ against r_2 and $10 > 2$ against r_3. So c_1 is **strictly dominated** by c_2. Eliminate c_1.

Step 4 (Row player): With r_2, r_3 and columns c_2, c_3:

$$A'' = \begin{pmatrix} 3 & 1 \\ 6 & 7 \end{pmatrix} \tag{3.32}$$

Compare $r_2 = (3, 1)$ and $r_3 = (6, 7)$: r_3 strictly dominates r_2 on both entries. Eliminate r_2.

Step 5 (Column player): With only r_3 and columns c_2, c_3:

$$B''' = (10 \;\; 12) \tag{3.33}$$

c_3 gives $12 > 10$, so c_2 is strictly dominated by c_3. Eliminate c_2.

Predicted outcome: (r_3, c_3).

Game 3:

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} \tag{3.34}$$

Step 1 (Column player): Compare columns for the column player:

- c_1 gives $(2, 2)$, c_2 gives $(1, 3)$, c_3 gives $(3, 2)$.

Check c_1 vs c_3: c_3 gives $3 > 2$ against r_1 but $2 = 2$ against r_2. So c_1 is **weakly dominated** by c_3. Eliminate c_1.

Step 2 (Row player): With c_2 and c_3 remaining:

$$A' = \begin{pmatrix} 3 & 2 \\ 1 & 3 \end{pmatrix} \tag{3.35}$$

Compare $r_1 = (3, 2)$ and $r_2 = (1, 3)$: neither dominates the other (row 1 is better against c_2, row 2 is better against c_3). We cannot eliminate further.

Result: Iterated elimination of weakly dominated strategies reduces the game to:

$$A' = \begin{pmatrix} 3 & 2 \\ 1 & 3 \end{pmatrix} \qquad B' = \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix} \tag{3.36}$$

No unique prediction is obtained. Further analysis (e.g. best responses) is required.

Game 4:

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{3.37}$$

Row player: Compare $r_1 = (3, -1)$ and $r_2 = (2, 7)$. Neither dominates: $r_1 > r_2$ against c_1, but $r_2 > r_1$ against c_2.

Column player: Compare $c_1 = (-3, 1)$ and $c_2 = (1, -6)$ for column payoffs. Neither dominates: $c_2 > c_1$ against r_1, but $c_1 > c_2$ against r_2.

Result: No strategies can be eliminated. Iterated elimination does not reduce this game.

Solution 3.2: Solution to Exercise 2

We identify all best response actions for each game in Exercise 1 and attempt to predict rational outcomes.

Game 1:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix} \tag{3.38}$$

Underline best responses (maximum in each column for the row player, maximum in each row for the column player):

$$A = \begin{pmatrix} \underline{2} & \underline{1} \\ 1 & \underline{1} \end{pmatrix} \qquad B = \begin{pmatrix} \underline{1} & \underline{1} \\ 1 & \underline{3} \end{pmatrix} \tag{3.39}$$

- Row player: r_1 is the unique best response to c_1; both r_1 and r_2 are best responses to c_2.
- Column player: Both c_1 and c_2 are best responses to r_1 (payoff tied at 1); c_2 is the unique best response to r_2.

Pairs where both are mutual best responses: (r_1, c_1), (r_1, c_2) and (r_2, c_2). There are predicted rational outcomes.

Game 2:

$$A = \begin{pmatrix} 2 & 1 & 3 & 17 \\ 27 & 3 & 1 & 1 \\ 4 & 6 & 7 & 18 \end{pmatrix} \qquad B = \begin{pmatrix} 11 & 9 & 10 & 22 \\ 0 & 1 & 1 & 0 \\ 2 & 10 & 12 & 0 \end{pmatrix} \tag{3.40}$$

Best response underlines:

$$A = \begin{pmatrix} 2 & 1 & 3 & 17 \\ \underline{27} & 3 & 1 & 1 \\ 4 & \underline{6} & \underline{7} & \underline{18} \end{pmatrix} \qquad B = \begin{pmatrix} 11 & 9 & 10 & \underline{22} \\ 0 & \underline{1} & \underline{1} & 0 \\ 2 & 10 & \underline{12} & 0 \end{pmatrix} \tag{3.41}$$

- Row player: r_2 is best response to c_1; r_3 is best response to c_2, c_3, and c_4.
- Column player: c_4 is best response to r_1; c_2, c_3 are all best responses to r_2; c_3 is best response to r_3.

The unique mutual best response pair is (r_3, c_3), confirming the predicted outcome from iterated elimination.

Game 3:

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} \tag{3.42}$$

Best response underlines:

$$A = \begin{pmatrix} \underline{3} & \underline{3} & 2 \\ 2 & 1 & \underline{3} \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & \underline{3} \\ 2 & \underline{3} & 2 \end{pmatrix} \tag{3.43}$$

- Row player: r_1 is best response to c_1 and c_2; r_2 is best response to c_3.
- Column player: c_2 is best response to r_2; c_3 is best response to r_1.

There are no pairs where both are mutual best responses:

The approach of pure strategy best responses fails to fully determine the outcome.

Game 4:

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{3.44}$$

Best response underlines:

$$A = \begin{pmatrix} \underline{3} & -1 \\ 2 & \underline{7} \end{pmatrix} \qquad B = \begin{pmatrix} -3 & \underline{1} \\ \underline{1} & -6 \end{pmatrix} \tag{3.45}$$

- Row player: r_1 is best response to c_1; r_2 is best response to c_2.
- Column player: c_2 is best response to r_1; c_1 is best response to r_2.

No action profile has both entries underlined simultaneously. Best response reasoning fails to determine the outcome.

Solution 3.3: Solution to Exercise 3

1. **Players and action sets:**

The players are Country 1 and Country 2 (the row and column players). The action sets are:

$$\mathcal{A}_1 = \mathcal{A}_2 = \{\text{Peace},\ \text{Attack}\} \tag{3.46}$$

2. **Normal form with utilities as negative of casualties:**

$$M_r = \begin{pmatrix} 0 & -15000 \\ -13000 & -10000 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0 & -13000 \\ -15000 & -10000 \end{pmatrix} \tag{3.47}$$

Rows and columns correspond to (Peace, Attack). When Country 1 attacks and Country 2 is peaceful, Country 1 suffers 13000 casualties (from retaliation) and Country 2 suffers 15000 casualties.

3. **Expected utility plots:**

Let $\sigma_2 = (y, 1-y)$ where y is the probability Country 2 plays Peace. Country 1 plays a pure action (Peace or Attack); its expected utilities are:

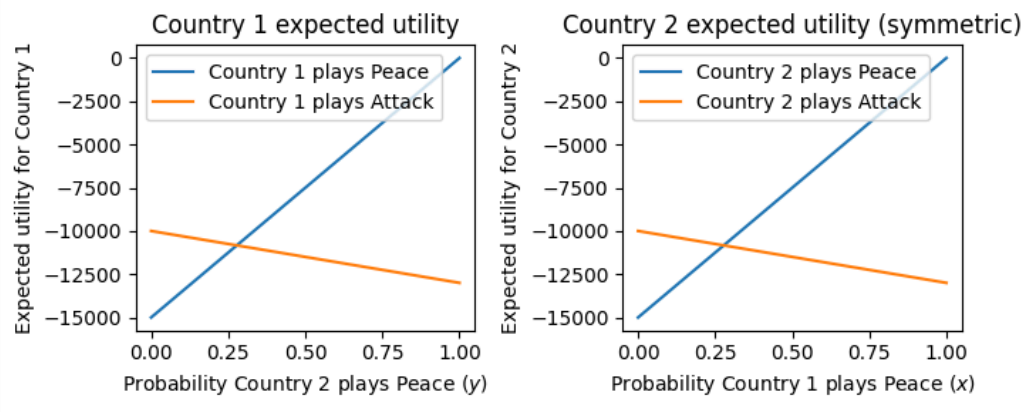
$$u_1(\text{Peace},\ \sigma_2) = 0 \cdot y + (-15000)(1-y) = -15000 + 15000y \tag{3.48}$$

$$u_1(\text{Attack},\ \sigma_2) = (-13000)y + (-10000)(1-y) = -3000y - 10000 \tag{3.49}$$

By symmetry the same expressions give Country 2's utilities when Country 1 plays the generic strategy $\sigma_1 = (x, 1-x)$.

```python
import matplotlib.pyplot as plt
import numpy as np

y = np.linspace(0, 1, 100)

u_peace  = -15000 + 15000 * y   # utility of playing Peace
u_attack = -3000 * y - 10000    # utility of playing Attack

plt.figure(figsize=(7, 3))

plt.subplot(1, 2, 1)
plt.plot(y, u_peace,  label="Country 1 plays Peace")
plt.plot(y, u_attack, label="Country 1 plays Attack")
plt.xlabel("Probability Country 2 plays Peace ($y$)")
plt.ylabel("Expected utility for Country 1")
plt.title("Country 1 expected utility")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(y, u_peace,  label="Country 2 plays Peace")
plt.plot(y, u_attack, label="Country 2 plays Attack")
plt.xlabel("Probability Country 1 plays Peace ($x$)")
```

```
plt.ylabel("Expected utility for Country 2")
plt.title("Country 2 expected utility (symmetric)")
plt.legend()

plt.tight_layout()
```



4. **Best responses:**

Peace is preferred to Attack when:

$$-15000 + 15000y > -3000y - 10000 \implies 18000y > 5000 \implies y > \frac{5}{18} \approx 0.28 \quad (3.50)$$

For Country 1:

- When Country 2 plays Peace with probability $y > \frac{5}{18}$: **Peace** is the best response.
- When Country 2 plays Peace with probability $y < \frac{5}{18}$: **Attack** is the best response.
- At $y = \frac{5}{18}$: Country 1 is indifferent.

By symmetry the same thresholds apply to Country 2 (replacing y with x). Attack is therefore not universally dominant: Peace is the best response when the opponent is sufficiently likely to be peaceful.

Solution 3.4: Solution to Exercise 4

1. **Plausible utility table:**

We define utilities to reflect the trade-off between market share and margin. Let the total market profit be normalised so that equal pricing splits it. A lower price steals customers but earns less per unit; a higher price loses customers but earns more per unit (up to a point).

A plausible assignment (in arbitrary units) for Shop 1 (row) and Shop 2 (column):

$$M_r = \begin{pmatrix} 3 & 5 & 6 \\ 1 & 3 & 5 \\ 0 & 1 & 3 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 1 & 0 \\ 5 & 3 & 1 \\ 6 & 5 & 3 \end{pmatrix} \quad (3.51)$$

where rows/columns correspond to (Low, Medium, High). When prices are equal, each shop earns 3. When Shop 1 undercuts Shop 2, Shop 1 earns more and Shop 2 earns less.

2. **Normal form:** The game is represented by the two matrices M_r and M_c above.

3. **Iterated elimination of dominated strategies:**

For the row player, compare High $(0, 1, 3)$ to Medium $(1, 3, 5)$: Medium gives strictly higher payoff against all column actions. High is **strictly dominated** by Medium. Eliminate High for the row player.

By symmetry (the game is symmetric under reflection), High is also **strictly dominated** for the column player. Eliminate the High column.

After elimination:

$$M_r{}' = \begin{pmatrix} 3 & 5 \\ 1 & 3 \end{pmatrix} \qquad M_c{}' = \begin{pmatrix} 3 & 1 \\ 5 & 3 \end{pmatrix} \tag{3.52}$$

Now compare Low $(3, 5)$ to Medium $(1, 3)$ for the row player: Low strictly dominates Medium (in the reduced game). Eliminate Medium.

By symmetry eliminate Medium for the column player.

After full elimination: both shops play **Low**.

4. **Best response pairs:**

After iterated elimination, the only remaining action for both players is Low. (Low, Low) is a pair of best responses.

Solution 3.5: Solution to Exercise 5

1. **Utility matrix construction:**

Let Worker 1 be the row player and Worker 2 be the column player. Actions are Documentation (D), Coding (C), Debugging (Db). Utilities reflect:

- If same task: penalty for coordination overhead, base utility reduced.
- If different tasks: bonus for complementary work.
- Each worker has individual skill strengths: Worker 1 is better at Coding, Worker 2 is better at Debugging.

$$M_r = \begin{pmatrix} 1 & 3 & 4 \\ 4 & 2 & 5 \\ 3 & 4 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & 4 & 3 \\ 3 & 2 & 4 \\ 5 & 4 & 1 \end{pmatrix} \tag{3.53}$$

Rows/columns correspond to (Documentation, Coding, Debugging). Diagonal entries are lower (same task, coordination overhead). Off-diagonal entries are higher (different tasks, complementary productivity).

2. **Normal form:** The game is represented by M_r and M_c above.
3. **Dominated strategies:**

No pure strategy is strictly or weakly dominated.

4. **Best responses:**

Underline row player's best responses (column-wise maxima) and column player's best responses (row-wise maxima):

$$M_r = \begin{pmatrix} 1 & 3 & 4 \\ \underline{4} & 2 & \underline{5} \\ 3 & \underline{4} & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & \underline{4} & 3 \\ 3 & 2 & \underline{4} \\ \underline{5} & 4 & 1 \end{pmatrix} \tag{3.54}$$

The unique mutual best response pair: (r_2, c_3): Worker 1 codes and Worker 2 debugs.

4. Zero-Sum Games

In some strategic interactions, every gain for one player comes at an equal cost to another. This chapter studies this important special case, showing how minimax optimisation and linear programming together characterise optimal play when interests are directly opposed.

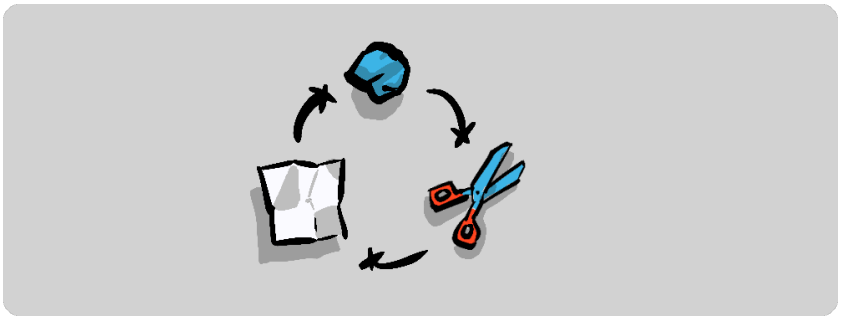


Figure 4.1: Rock, Paper, and Scissors chase one another in a cycle, with no single action dominant. Such cyclic advantage is characteristic of zero-sum games, where one player's gain is exactly the other player's loss.

4.1 Motivating Example

Consider the standard **Rock-Paper-Scissors** game The payoff matrix for the row player is:

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \tag{4.1}$$

We are going to modify the game to reflect the fact that the row player enjoys winning and hates losing with Paper more than any other action:

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 2 & 0 & -2 \\ -1 & 1 & 0 \end{pmatrix} \tag{4.2}$$

The column player feels equally strongly about Paper-related outcomes.

Is there a way for the **row player** to choose a strategy that guarantees a certain minimum expected payoff, regardless of how the column player responds?

4.2 Theory

This chapter will consider a specific subset of Normal Form Games.

4.2.1 Definition: Zero Sum Game

A two player normal form game with payoff matrices $(M_r, M_c) \in \mathbb{R}^{(m \times n)^2}$ is called zero sum if and only if:

$$M_r + M_c = 0 \tag{4.3}$$

In the case of a zero sum game we will use the convention of defining it with:

$$M = M_r \tag{4.4}$$

and implying $M_c = -M$.

4.2.2 Definition: the min-max and max-min strategies

Given a zero-sum game defined by a payoff matrix $M \in \mathbb{R}^{m \times n}$ and a strategy $y \in \mathbb{R}^n$ for the **column player**, the **row player** seeks a best response strategy $x \in \mathbb{R}^m$ that maximises their expected payoff:

$$\max_{x \in \mathcal{A}_1} xMy^T \tag{4.5}$$

This corresponds to choosing the rows of M that yields the highest expected value under the strategy y, i.e.,

$$\max_{i \leq m} \left(My^T\right)_i \tag{4.6}$$

The column player, by selecting y, can influence the upper bound v of this maximum. Since the game is zero-sum, the column player will aim to choose y to make this upper bound v as small as possible.

Hence,

$$\max_{x \in \mathcal{A}_1} xMy^T = \max_{i \leq m} \left(My^T\right)_i = \min\{v \in \mathbb{R} \mid My^T \leq 1v\} \tag{4.7}$$

The **min-max strategy** y for the column player is the solution to the following optimisation problem (referred to as a **linear program**):

$$\begin{aligned} \min_{y,v} \quad & v \\ \text{subject to} \quad & My^T \leq 1v \\ & y \in \mathcal{A}_2 \end{aligned} \tag{4.8}$$

In this formulation, v is the **min-max value** of the game.

The corresponding **max-min strategy** x for the row player solves the following linear program:

$$\begin{aligned} \max_{x,u} \quad & u \\ \text{subject to} \quad & xM \geq 1u \\ & x \in \mathcal{A}_1 \end{aligned} \tag{4.9}$$

In this case, u is the **max-min value** of the game.

4.2.2.1 Example: Max-min strategy for modified Rock-Paper-Scissors

For (4.2), the **max-min strategy** x for the row player satisfies the following linear program:

$$\begin{aligned} \max_{x,u} \quad & u \\ \text{subject to} \quad & 2x_2 - x_3 \geq u \\ & -x_1 + x_3 \geq u \\ & x_1 - 2x_2 \geq u \\ & x_1 + x_2 + x_3 = 1 \\ & x_i \geq 0 \quad \text{for all } i \in \{1, 2, 3\} \end{aligned} \tag{4.10}$$

4.2.2.2 Example: Max-min strategy for Matching Pennies

For the matching pennies game with payoff matrix:

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{4.11}$$

the **max-min strategy** x for the row player satisfies the following linear program:

$$
\begin{aligned}
\max_{x,u} \quad & u \\
\text{subject to} \quad & x_1 - x_2 \geq u \\
& -x_1 + x_2 \geq u \\
& x_1 + x_2 = 1 \\
& x_i \geq 0 \quad \text{for all } i \in \{1, 2\}
\end{aligned}
\tag{4.12}
$$

Given that $x_1 + x_2 = 1$, this linear program corresponds to:

$$
\begin{aligned}
\max_{x_1,u} \quad & u \\
\text{subject to} \quad & 2x_1 - 1 \geq u \\
& -2x_1 + 1 \geq u \\
& 0 \leq x_1 \leq 1
\end{aligned}
\tag{4.13}
$$

These constraints can be rewritten as:

$$
\begin{aligned}
x_1 &\geq \frac{1+u}{2} \\
x_1 &\leq \frac{1-u}{2} \\
0 &\leq x_1 \leq 1
\end{aligned}
\tag{4.14}
$$

This implies:

$$
\frac{1+u}{2} \leq x_1 \leq \frac{1-u}{2} \tag{4.15}
$$

which leads to:

$$
\frac{1+u}{2} \leq \frac{1-u}{2} \quad \Rightarrow \quad u \leq -u \tag{4.16}
$$

This inequality holds only when $u = 0$. When $u = 0$, the constraints reduce to:

$$
\frac{1}{2} \leq x_1 \leq \frac{1}{2} \tag{4.17}
$$

yielding the unique solution $x_1 = \frac{1}{2}$.

Thus, the **max-min strategy** is:

$$
x = \left(\frac{1}{2}, \frac{1}{2}\right) \tag{4.18}
$$

4.2.3 Theorem: The minimax theorem

The minimax theorem (Neumann, 1928) states that if there exists optimal values of the:

1. *max-min* value u and the *max-min* strategy x.
2. *min-max* value v and the *min-max* strategy y.

then $u = v$.

The proof which uses the **linear program duality theorem** is omitted from this book but can be found in (Vanderbei, 2010).

Note that this answers the question posed at the end of Motivating Example through a choice of strategy the row player can ensure they obtain the value of the game which is equal to the max-min value and the min-max value.

In the next section we will start to introduce practical tools with which to do that.

4.2.4 Definition: Standard Form Linear program

A standard form of the linear program can be written which more readily will allow us to use Integer Pivoting.

In a zero-sum game, given a row player payoff matrix M with m rows and n columns, the following linear program yields the **max-min strategy** and the **value** of the game:

$$\min_{x \in \mathbb{R}^{(m+1)\times 1}} cx \tag{4.19}$$

subject to:

$$\begin{aligned} M_{\text{ub}} x &\leq b_{\text{ub}} \\ M_{\text{eq}} x &= b_{\text{eq}} \\ x_i &\geq 0 \quad \text{for } i \leq m \end{aligned} \tag{4.20}$$

The coefficients in this linear program are defined as:

$$\begin{aligned} c &= \left(\underbrace{0, \ldots, 0}_{m}, -1\right) && \text{where } c \in \{0,1\}^{1\times(m+1)} \\ M_{\text{ub}} &= \begin{pmatrix} (-M^T)_{11} & \ldots & (-M^T)_{1m} & 1 \\ \vdots & \ddots & \vdots & 1 \\ (-M^T)_{n1} & \ldots & (-M^T)_{nm} & 1 \end{pmatrix} && M_{\text{ub}} \in \mathbb{R}^{n\times(m+1)} \\ b_{\text{ub}} &= \left(\underbrace{0, \ldots, 0}_{n}\right)^T && b_{\text{ub}} \in \mathbb{R}^{n\times 1} \\ M_{\text{eq}} &= \left(\underbrace{1, \ldots, 1}_{m}, 0\right) && M_{\text{eq}} \in \{0,1\}^{1\times(m+1)} \\ b_{\text{eq}} &= 1 \end{aligned} \tag{4.21}$$

4.2.4.1 Example: Standard form for the Modified Rock Paper Scissors Game

For the modified Rock-Paper-Scissors game, the corresponding coefficients are:

$$\begin{aligned} c &= (0, 0, 0, -1) \\ M_{\text{ub}} &= \begin{pmatrix} 0 & -2 & 1 & 1 \\ 1 & 0 & -1 & 1 \\ -1 & 2 & 0 & 1 \end{pmatrix} \\ b_{\text{ub}} &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ M_{\text{eq}} &= (1, 1, 1, 0) \\ b_{\text{eq}} &= 1 \end{aligned} \tag{4.22}$$

4.2.5 Definition: Tableau for Zero-Sum Game

Given a zero-sum game with payoff matrix $M \in \mathbb{R}m \times n$ the standard form linear program: can be represented by the following **initial tableau**:

$$T = \begin{pmatrix} x_1 & \dots & x_m & v & s_1 & \dots & s_n & b \\ (-M^T)^{11} & \dots & (-M^T)_{1m} & 1 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & 1 & 0 & \ddots & 0 & 0 \\ (-M^T)_{n1} & \dots & (-M^T)_{nm} & 1 & 0 & \dots & 1 & 0 \\ 1 & \dots & 1 & 0 & 0 & \dots & 1 & 1 \\ 0 & \dots & 0 & -1 & 0 & \dots & 0 & 0 \end{pmatrix} \tag{4.23}$$

Here, slack variables have been introduced to convert inequalities into equalities. The tableau is arranged with columns for the decision variables $x_1, \dots, x_m$, the game value variable v, slack variables $s_1, \dots, s_n$, and the right-hand side.

Important

The final row of the tableau has been somewhat artifically added: it does not correspond to any specific part of (4.21). However this is an important addition to the tableau as it will indicate two things:

- Which column to choose to pivot in other words which variable to make non-basic.
- When to stop integer pivoting.

Given a Tableau, we choose to pivot the column with the lowest negative value in the final row: this corresponds to the variable that will reduce the objective function the most. When all values in the final row are non-negative we stop integer pivoting.

We proceed by performing integer pivoting to move from one basic feasible solution to another, reducing the objective function at each step until optimality is reached.

4.2.5.1 Example: Integer Pivoting for Modified Rock-Paper-Scissors

We now solve the modified Rock-Paper-Scissors game using tableaux. Recall that the standard form coefficients are:

$$\begin{aligned} c &= (0, 0, 0, -1) \\ M_{\text{ub}} &= \begin{pmatrix} 0 & -2 & 1 & 1 \\ 1 & 0 & -1 & 1 \\ -1 & 2 & 0 & 1 \end{pmatrix} \\ b_{\text{ub}} &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ M_{\text{eq}} &= (1, 1, 1, 0) \\ b_{\text{eq}} &= 1 \end{aligned} \tag{4.24}$$

To construct the tableau, we introduce slack variables s_1, s_2, s_3 for the inequality constraints, and denote the game value variable by $v = x_5$.

4.2.5.2 Initial Tableau

The initial tableau is:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 0 & -2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array} \tag{4.25}$$

The last row is the objective function: minimising v. We begin by identifying the entering variable with the most negative coefficient in the objective row, which is v.

4.2.5.3 Pivot 1: Entering variable v

To pivot on v, we look at the positive entries in the v column (rows 1–3). All are 1, so we apply the **ratio test**:

- Row 1: $0/1 = 0$
- Row 2: $0/1 = 0$
- Row 3: $0/1 = 0$

Ties are broken arbitrarily. Suppose we choose row 1: we subtract appropriate multiples of this row from all others to eliminate v from those rows:

- Row 2 ← Row 2 - Row 1
- Row 3 ← Row 3 - Row 1
- Objective row = Objective + Row 1

After row operations:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 0 & -2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & -2 & 0 & -1 & 1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \tag{4.26}$$

4.2.5.4 Pivot 2: Entering variable x_2

Next, inspect the objective row. The most negative coefficient is –2 for x_2.

Apply ratio test on rows with positive x_2 entries:

- Row 2: $0/2 = 0$
- Row 3: $0/4 = 0$
- Row 4: $1/1 = 1$

Choose Row 2 (arbitrary tie-break).

Pivot on entry in row 2, column x_2 and eliminate x_2 from other rows. After computations, we get:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 2 & 0 & -2 & 2 & 0 & 2 & 0 & 0 \\ 1 & 2 & -2 & 0 & -1 & 1 & 0 & 0 \\ -6 & 0 & 6 & 0 & 2 & -4 & 2 & 0 \\ 1 & 0 & 4 & 0 & 1 & -1 & 0 & 2 \\ 2 & 0 & -2 & 0 & 0 & 2 & 0 & 0 \end{array} \tag{4.27}$$

4.2.5.5 Pivot 3: Entering variable x_3

Continue inspecting the objective row. The most negative is now –2 for x_3, so it enters. Apply ratio test among positive entries in column x_3.

- Row 3: $0/6 = 0$

There is a single candidate: pivot on row 3. Eliminate x_3 from other rows. After computations we get:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 0 & 0 & 0 & 12 & 4 & 4 & 4 & 0 \\ -6 & 12 & 0 & 0 & -2 & -2 & 4 & 0 \\ -6 & 0 & 6 & 0 & 2 & -4 & 2 & 0 \\ 30 & 0 & 0 & 0 & -2 & 10 & -8 & 12 \\ 0 & 0 & 0 & 0 & 4 & 4 & 4 & 0 \end{array} \tag{4.28}$$

We now set the non-basic variables to 0 and read the equations for the non-basic variables:

$$\begin{aligned} s_1 &= 0 \\ s_2 &= 0 \\ s_3 &= 0 \\ 12v &= 0 \\ -6x_1 + 12x_2 &= 0 \\ -6x_1 + 6x_3 &= 0 \\ 30x_1 &= 12 \end{aligned} \tag{4.29}$$

This gives:

$$x = (12/30, 6/30, 12/30) \qquad v = 0 \tag{4.30}$$

Thus, the **max-min strategy** is:

$$x = \left(\frac{2}{5}, \frac{1}{5}, \frac{2}{5}\right) \tag{4.31}$$

4.3 Exercises

Exercise 4.1:

Obtain the coefficients of the standard form linear system for the zero-sum games with the following payoff matrices:

$$M = \begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix} \tag{4.32}$$

$$M = \begin{pmatrix} -1 & -1 \\ -1 & 3 \end{pmatrix} \tag{4.33}$$

$$M = \begin{pmatrix} 2 & 1 & -3 \\ -3 & -1 & 3 \end{pmatrix} \tag{4.34}$$

$$M = \begin{pmatrix} 3 & -2 & 0 \\ -3 & 0 & 3 \\ 0 & 2 & -5 \end{pmatrix} \tag{4.35}$$

Exercise 4.2:

For the matching pennies game with payoff matrix:

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{4.36}$$

1. Use integer pivoting to confirm that the max-min strategy is $x = (1/2, 1/2)$.

2. By letting $M = -M^T$, or otherwise, obtain the min-max strategy for the column player.
3. Use the Best Response Condition to confirm your calculations.

Exercise 4.3:

Obtain the max-min strategy for the standard game of Rock Paper Scissors defined by:

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \tag{4.37}$$

4.4 Programming

4.4.1 Solve linear programs using Scipy

The `scipy` library provides functionality to solve a linear program in standard form.

We begin by creating the various matrices and vectors: M_{ub}, M_{eq}, b_{ub}, b_{eq}, and c:

```
import numpy as np

M = np.array(
    [
        [0, -1, 1],
        [2, 0, -2],
        [-1, 1, 0],
    ]
)
M_ub = np.hstack((-M.T, [[1], [1], [1]]))
M_eq = np.array(([[1, 1, 1, 0]]))
b_ub = np.array(
    [
        [0],
        [0],
        [0],
    ]
)
b_eq = 1
c = np.array([0, 0, 0, -1])
```

Now we can pass these to `scipy.optimize.linprog`:

```
import scipy.optimize

res = scipy.optimize.linprog(
    c=c,
    A_ub=M_ub,
    b_ub=b_ub,
    A_eq=M_eq,
    b_eq=b_eq,
)
print(res)
```

```
       message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
       success: True
        status: 0
           fun: 0.0
             x: [ 4.000e-01  2.000e-01  4.000e-01 -0.000e+00]
           nit: 0
         lower:  residual: [ 4.000e-01  2.000e-01  4.000e-01 -0.000e+00]
                marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
         upper:  residual: [       inf        inf        inf        inf]
                marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
         eqlin:  residual: [ 0.000e+00]
                marginals: [-0.000e+00]
       ineqlin:  residual: [ 0.000e+00  0.000e+00  0.000e+00]
                marginals: [-3.333e-01 -3.333e-01 -3.333e-01]
 mip_node_count: 0
 mip_dual_bound: 0.0
       mip_gap: 0.0
```

This returns the full output of the optimisation. The min-max strategy is contained in all but the last entry of `res.x`:

```
print(f"Min-max strategy: {res.x[:-1]}")
```

```
Min-max strategy: [0.4 0.2 0.4]
```

The last entry of `res.x` gives the value of the game:

```
print(f"Value of the game: {res.x[-1]}")
```

```
Value of the game: -0.0
```

4.4.2 Obtain min-max and max-min strategies using Nashpy

`nashpy` can be used to directly obtain the min-max and max-min strategies:

```
import nashpy as nash

game = nash.Game(M, -M)
print(game.linear_program())
```

```
(array([0.4, 0.2, 0.4]), array([0.33333333, 0.33333333, 0.33333333]))
```

4.4.3 Obtain min-max and max-min strategies using Gambit

`Gambit` can be used to directly obtain the min-max and max-min strategies. We start by creating a pygambit game from arrays:

```
import pygambit as gbt

game = gbt.Game.from_arrays(M, -M)
print(game)
```

```
Game(title='Untitled strategic game')
```

Now we can solve the underlying linear program:

```
print(gbt.nash.lp_solve(game))
```

```
NashComputationResult(method='lp', rational=True, use_strategic=False,
equilibria=[[[Rational(2, 5), Rational(1, 5), Rational(2, 5)], [Rational(1, 3),
Rational(1, 3), Rational(1, 3)]]], parameters={})
```

4.5 Notable Research

The foundations of zero-sum game theory and its connection to linear programming emerged from a convergence of ideas in mathematics, economics, and operations research during the mid-20th century.

The **minimax theorem** was first proven by John von Neumann in 1928 (Neumann, 1928). This landmark result, stating that every finite, two-player zero-sum game has a value and optimal strategies, was later generalised in 1944 (Neumann & Morgenstern, 1944).

The **minimax theorem** does not necessarily only apply to zero-sum games but in fact applies to any constant sum game where $M_r + M_c = K$ for some constant K. An example of this is shown in (Chiappori et al., 2002) where penalty kicks are modelled and the payoff matrices correspond to the probability of scoring (or for the column player saving) a penalty.

Until the work of (Nash Jr, 1950) the minimax theorem was the main solution concept in game theory. For his foundational work on equilibrium in non-cooperative games, John Nash was awarded the Nobel Prize in Economic Sciences in 1994, shared with John Harsanyi and Reinhard Selten. His contributions form the cornerstone of non-cooperative game theory.

4.6 Conclusion

This chapter introduced zero-sum games, where one player's gain is precisely balanced by the other's loss. We explored the foundational minimax theorem, the max-min and min-max strategies, and showed how linear programming provides a practical and elegant way to compute optimal strategies.

The central insight of this chapter is the equivalence between solving a zero-sum game and solving a pair of dual linear programs. This connection allows us to apply tools from optimisation, such as tableau methods and integer pivoting, to find equilibrium strategies.

Table 4.1 summarises the two central linear programs seen in this chapter.

Problem	Player	Objective	Constraints
Max-min LP	Row player	Maximise u	$xM \geq 1u, x \in \mathcal{A}_1$
Min-max LP	Column player	Minimise v	$My^T \leq 1v, y \in \mathcal{A}_2$

Table 4.1: The main linear programs for Zero Sum Game

Attention

The **value of a zero-sum game** is the unique number that both players can guarantee for themselves through optimal play, and it can be computed by solving either the max-min or min-max linear program.

4.7 Solutions

Solution 4.1: Solution to Exercise 1

We obtain the standard form linear program coefficients for each game. Recall the standard form uses:

$$c = (0, ..., 0, -1), \quad M_{\text{ub}} = (-M^T \ \ 1), \quad b_{\text{ub}} = 0, \quad M_{\text{eq}} = (1, ..., 1, 0), \quad b_{\text{eq}} = 1 \tag{4.38}$$

1. $M = \begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix}$

This is a 2×2 game so the decision vector is (x_1, x_2, v).

$$-M^T = -\begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix}^T = -\begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} -3 & 1 \\ 1 & -2 \end{pmatrix} \tag{4.39}$$

$$c = (0, 0, -1) \tag{4.40}$$

$$M_{\text{ub}} = \begin{pmatrix} -3 & 1 & 1 \\ 1 & -2 & 1 \end{pmatrix} \qquad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{4.41}$$

$$M_{\text{eq}} = (1, 1, 0) \qquad b_{\text{eq}} = 1 \tag{4.42}$$

---

2. $M = \begin{pmatrix} -1 & -1 \\ -1 & 3 \end{pmatrix}$

$$-M^T = \begin{pmatrix} 1 & 1 \\ 1 & -3 \end{pmatrix} \tag{4.43}$$

$$c = (0, 0, -1) \tag{4.44}$$

$$M_{\text{ub}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -3 & 1 \end{pmatrix} \qquad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{4.45}$$

$$M_{\text{eq}} = (1, 1, 0) \qquad b_{\text{eq}} = 1 \tag{4.46}$$

---

3. $M = \begin{pmatrix} 2 & 1 & -3 \\ -3 & -1 & 3 \end{pmatrix}$

This is a 2×3 game. The row player has 2 actions, so the decision vector is (x_1, x_2, v).

$$-M^T = -\begin{pmatrix} 2 & -3 \\ 1 & -1 \\ -3 & 3 \end{pmatrix} = \begin{pmatrix} -2 & 3 \\ -1 & 1 \\ 3 & -3 \end{pmatrix} \tag{4.47}$$

$$c = (0, 0, -1) \tag{4.48}$$

$$M_{\text{ub}} = \begin{pmatrix} -2 & 3 & 1 \\ -1 & 1 & 1 \\ 3 & -3 & 1 \end{pmatrix} \qquad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.49}$$

$$M_{\text{eq}} = (1, 1, 0) \qquad b_{\text{eq}} = 1 \tag{4.50}$$

---

4. $M = \begin{pmatrix} 3 & -2 & 0 \\ -3 & 0 & 3 \\ 0 & 2 & -5 \end{pmatrix}$

This is a 3×3 game. The decision vector is (x_1, x_2, x_3, v).

$$-M^T = -\begin{pmatrix} 3 & -3 & 0 \\ -2 & 0 & 2 \\ 0 & 3 & -5 \end{pmatrix} = \begin{pmatrix} -3 & 3 & 0 \\ 2 & 0 & -2 \\ 0 & -3 & 5 \end{pmatrix} \tag{4.51}$$

$$c = (0, 0, 0, -1) \tag{4.52}$$

$$M_{\text{ub}} = \begin{pmatrix} -3 & 3 & 0 & 1 \\ 2 & 0 & -2 & 1 \\ 0 & -3 & 5 & 1 \end{pmatrix} \qquad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.53}$$

$$M_{\text{eq}} = (1, 1, 1, 0) \qquad b_{\text{eq}} = 1 \tag{4.54}$$

Solution 4.2: Solution to Exercise 2

Recall the Matching Pennies payoff matrix:

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{4.55}$$

1. **Integer pivoting to confirm max-min strategy $x = (1/2, 1/2)$:**

The standard form coefficients are:

$$c = (0, 0, -1), \quad M_{\text{ub}} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}, \quad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad M_{\text{eq}} = (1, 1, 0), \quad b_{\text{eq}} = 1 \tag{4.56}$$

The initial tableau (introducing slack variables s_1, s_2) is:

$$\begin{array}{cccccc} x_1 & x_2 & v & s_1 & s_2 & b \\ -1 & 1 & 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{array} \tag{4.57}$$

Pivot 1: Entering variable v (most negative objective coefficient $= -1$).

All entries in the v column among constraint rows are 1. Ratio test: all rows give $0/1 = 0$. Choose Row 1 as the pivot row.

- Row 2 $\leftarrow$ Row 2 $-$ Row 1
- Row 3 stays (Row 3 has no v entry)
- Objective row $\leftarrow$ Objective $+$ Row 1

$$\begin{array}{cccccc} x_1 & x_2 & v & s_1 & s_2 & b \\ -1 & 1 & 1 & 1 & 0 & 0 \\ 2 & -2 & 0 & -1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 & 0 & 0 \end{array} \tag{4.58}$$

Pivot 2: Entering variable x_1 (most negative objective coefficient $= -1$).

Positive entries in x_1 column: Row 2 (2) and Row 3 (1). Ratio test: Row 2 gives $0/2 = 0$, Row 3 gives $1/1 = 1$. Choose Row 2.

- Row 1 $\leftarrow$ Row 1 $\cdot\, 2 +$ Row 2

- Row 3 $\leftarrow$ Row 3 $\cdot$ 2 $-$ Row 2
- Objective $\leftarrow$ Objective $\cdot$ 2 $+$ Row 2

$$\begin{array}{cccccc} x_1 & x_2 & v & s_1 & s_2 & b \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 2 & -2 & 0 & -1 & 1 & 0 \\ 0 & 4 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array} \tag{4.59}$$

All objective row entries are non-negative: we stop. Setting non-basic variables ($s_1 = s_2 = 0$) and reading the system:

$$2v = 0 \implies v = 0 \tag{4.60}$$

$$2x_1 - 2x_2 = 0 \implies x_1 = x_2 \tag{4.61}$$

$$4x_2 = 2 \implies x_2 = \frac{1}{2} \tag{4.62}$$

Thus $x_1 = x_2 = 1/2$, confirming:

$$x = \left(\frac{1}{2}, \frac{1}{2}\right), \qquad v = 0 \tag{4.63}$$

2. Min-max strategy for the column player:

The column player's game is given by $-M^T$. Since M is antisymmetric ($M = -M^T$ for this game), the min-max LP for the column player is identical in structure. Setting $M' = -M^T$:

$$M' = -M^T = -\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.64}$$

By the identical symmetry of the problem, the min-max strategy for the column player is:

$$y = \left(\frac{1}{2}, \frac{1}{2}\right) \tag{4.65}$$

with min-max value $v = 0$. By the minimax theorem, $u = v = 0$.

3. Best Response Condition check:

With $x = (1/2, 1/2)$ and $y = (1/2, 1/2)$:

$$My^T = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{4.66}$$

Both entries equal $0 = \max_k \left(My^T\right)_k$, and the support of x is $\{1, 2\}$, so the best response condition is satisfied: x is a best response to y.

By symmetry:

$$xM = (1/2 \;\; 1/2) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = (0 \;\; 0) \tag{4.67}$$

Both entries of xM are equal, confirming y is a best response to x.

Let us confirm this with `nashpy``:

```
import numpy as np
import nashpy as nash
```

```
M = np.array([[1, -1], [-1, 1]])
game = nash.Game(M, -M)
game.linear_program()
```

```
(array([0.5, 0.5]), array([0.5, 0.5]))
```

Solution 4.3: Solution to Exercise 3

The standard Rock-Paper-Scissors payoff matrix is:

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \tag{4.68}$$

The standard form coefficients are:

$$c = (0, 0, 0, -1) \tag{4.69}$$

$$M_{\text{ub}} = \begin{pmatrix} 0 & -1 & 1 & 1 \\ 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 1 \end{pmatrix} \qquad b_{\text{ub}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.70}$$

$$M_{\text{eq}} = (1, 1, 1, 0), \qquad b_{\text{eq}} = 1 \tag{4.71}$$

Integer pivoting:

Initial tableau:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array} \tag{4.72}$$

Pivot 1: v enters (coefficient -1). Choose Row 1 (ratio $0/1 = 0$).

- Row 2 $\leftarrow$ Row 2 $-$ Row 1: $(1, 1, -2, 0, -1, 1, 0, 0)$
- Row 3 $\leftarrow$ Row 3 $-$ Row 1: $(-1, 2, -1, 0, -1, 0, 1, 0)$
- Objective $\leftarrow$ Objective $+$ Row 1: $(0, -1, 1, 0, 1, 0, 0, 0)$

$$\begin{array}{cccccccc} 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & -1 & 1 & 0 & 0 \\ -1 & 2 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \tag{4.73}$$

Pivot 2: x_2 enters (coefficient -1). Positive entries in x_2 column: Row 2 (1, ratio 0), Row 3 (2, ratio 0), Row 4 (1, ratio 1). Choose Row 2 (tie-break).

Pivot on $(2, x_2)$, entry = 1. Eliminate x_2 from other rows:

- Row 1 $\leftarrow$ Row 1 $+$ Row 2: $(1, 0, -1, 1, 0, 1, 0, 0)$
- Row 3 $\leftarrow$ Row 3 $- 2 \cdot$ Row 2: $(-3, 0, 3, 0, 1, -2, 1, 0)$

- Row 4 $\leftarrow$ Row 4 $-$ Row 2: $(0, 0, 3, 0, 1, -1, 0, 1)$
- Objective $\leftarrow$ Objective $+$ Row 2: $(1, 0, -1, 0, 0, 1, 0, 0)$

$$\begin{array}{cccccccc} 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & -2 & 0 & -1 & 1 & 0 & 0 \\ -3 & 0 & 3 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 3 & 0 & 1 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \end{array} \tag{4.74}$$

Pivot 3: x_3 enters (coefficient −1). Positive entries in x_3 column: Row 3 (3, ratio $0/3 = 0$), Row 4 (3, ratio $1/3$). Choose Row 3.

Pivot on $(3, x_3)$, entry = 3. Eliminate x_3 from other rows:

- Row 1 $\leftarrow 3 \cdot$ Row 1 $+$ Row 3: $(0, 0, 0, 3, 1, 1, 1, 0)$
- Row 2 $\leftarrow 3 \cdot$ Row 2 $+ 2 \cdot$ Row 3: $(-3, 3, 0, 0, -1, -1, 2, 0)$
- Row 4 $\leftarrow 3 \cdot$ Row 4 $- 3 \cdot$ Row 3: $(9, 0, 0, 0, 0, 3, -3, 3)$
- Objective $\leftarrow 3 \cdot$ Objective $+$ Row 3: $(0, 0, 0, 0, 1, 1, 1, 0)$

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & v & s_1 & s_2 & s_3 & b \\ 0 & 0 & 0 & 3 & 1 & 1 & 1 & 0 \\ -3 & 3 & 0 & 0 & -1 & -1 & 2 & 0 \\ -3 & 0 & 3 & 0 & 1 & -2 & 1 & 0 \\ 9 & 0 & 0 & 0 & 0 & 3 & -3 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array} \tag{4.75}$$

All entries in the objective row are non-negative: we stop. Setting non-basic variables ($s_1 = s_2 = s_3 = 0$) and reading the equations:

$$\begin{aligned} 3v &= 0 \\ -3x_1 + 3x_2 &= 0 \\ -3x_1 + 3x_3 &= 0 \\ 9x_1 &= 3 \end{aligned} \tag{4.76}$$

This gives $v = 0$, $x_1 = x_2 = x_3$, and $x_1 = 1/3$.

Thus:

$$x = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \qquad v = 0 \tag{4.77}$$

Let us confirm this with `nashpy`:

```
import numpy as np
import nashpy as nash

M = np.array([[0, -1, 1], [1, 0, -1], [-1, 1, 0]])
game = nash.Game(M, -M)
game.linear_program()
```

```
(array([0.33333333, 0.33333333, 0.33333333]),
 array([0.33333333, 0.33333333, 0.33333333]))
```

5. Nash Equilibrium

When every player's strategy is a best response to every other's, no one has an incentive to deviate unilaterally. This configuration is called a Nash equilibrium. This chapter formalises the concept, establishes its existence, and develops a systematic algorithm for computing it.

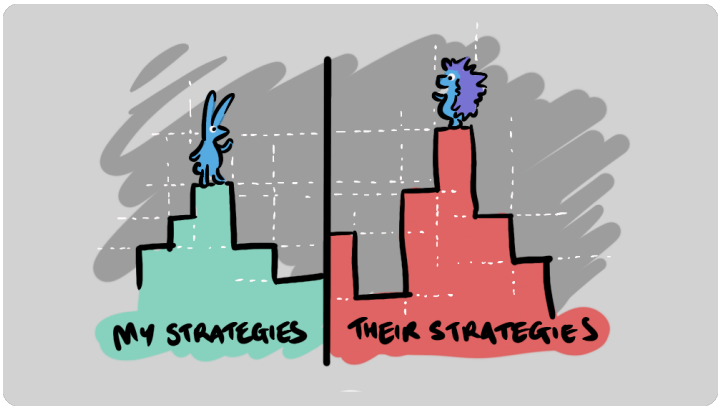


Figure 5.1: Each player climbs their own landscape of strategies, one shaped by what the other player does. At a Nash equilibrium both have reached the top of their respective hill: each is already standing on its best response to where the other stands, so neither has anywhere better to go.

5.1 Motivating Example

In the coordination Game (2.3), in how many situations do neither player have an incentive to **independently** change their strategy?

Neither player having a reason to change their strategy implies that both strategies are best responses to each other.

Recall that for the **Coordination game** is defined by:

$$M_r = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix} \tag{5.1}$$

If we consider strategies that only play a single action, there are two options for each strategy:

$$\sigma_1 \in \{(1,0),(0,1)\} \tag{5.2}$$

and:

$$\sigma_2 \in \{(1,0),(0,1)\} \tag{5.3}$$

We will inspect all four combinations:

- $\sigma_1 = (1,0)$ and $\sigma_2 = (1,0)$ corresponds to both players playing their first action, which gives: $u_r(\sigma_1,\sigma_2) = 3$ and $u_c(\sigma_1,\sigma_2) = 2$.

If the row player were to modify their strategy (while the column player stayed unchanged) to play the second action, their utility would decrease. Likewise, if the column player were to modify their strategy, their utility would also decrease.

- $\sigma_1 = (1,0)$ and $\sigma_2 = (0,1)$ corresponds to the row player playing their first action and the column player playing their second action, which gives: $u_r(\sigma_1,\sigma_2) = 1$ and $u_c(\sigma_1,\sigma_2) = 1$.

In this case, if either player were to move, their utility would increase.

- $\sigma_1 = (0,1)$ and $\sigma_2 = (1,0)$ corresponds to the row player playing their second action and the column player playing their first action, which gives: $u_r(\sigma_1,\sigma_2) = 0$ and $u_c(\sigma_1,\sigma_2) = 0$.

In this case, if either player were to move, their utility would increase.

- $\sigma_1 = (0, 1)$ and $\sigma_2 = (0, 1)$ corresponds to both players playing their second action, which gives: $u_r(\sigma_1, \sigma_2) = 2$ and $u_c(\sigma_1, \sigma_2) = 3$.

If the row player were to modify their strategy (while the column player stayed unchanged), their utility would decrease. Likewise, if the column player were to modify their strategy, their utility would also decrease.

Is there another pair of strategies that are best responses to each other and will such a pair always exist for any game?

5.2 Theory

A pair of strategies that are best responses to each other is a Nash equilibrium.

5.2.1 Definition: Nash Equilibria

In an N-player normal form game, a **Nash equilibrium** is a strategy profile $\tilde{s} = (\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_N)$ such that:

$$u_i(\tilde{s}) = \max_{\bar{s}_i \in \Delta(\mathcal{A}_i)} u_i(\bar{s}_i, \tilde{s}_{-i}) \quad \text{for all } i \tag{5.4}$$

Warning

In many game theory texts, Nash equilibria in normal form games are referred to as either pure strategy Nash equilibria, where each player chooses a single action with certainty, or mixed strategy Nash equilibria, where players randomise over multiple actions with positive probability.

5.2.2 Theorem: Existence of Nash Equilibrium

Every finite N-player normal form game has at least one Nash equilibrium.

5.2.2.1 Proof sketch

The proof, due to (Nash Jr, 1950), applies **Kakutani's fixed-point theorem** (Kakutani, 1941) to the best response correspondence. For each player i, define:

$$\beta_i(s_{-i}) = \arg\max_{\sigma_i \in \Delta(\mathcal{A}_i)} u_i(\sigma_i, s_{-i}) \tag{5.5}$$

Here $\beta_i(s_{-i})$ is the set of all mixed strategies for player i that are best responses to the opponents' strategies s_{-i}. The joint best response correspondence $\beta : \prod_i \Delta(\mathcal{A}_i) \to \prod_i \Delta(\mathcal{A}_i)$ satisfies the conditions of Kakutani's theorem (strategy spaces are compact and convex, and β has a closed graph and convex values), so a fixed point $\tilde{s}$ exists. Any fixed point satisfies $\tilde{s}_i \in \beta_i(\tilde{s}_{-i})$ for all i, which is precisely the Nash equilibrium condition.

Note

This theorem guarantees existence but not uniqueness. As seen in the coordination game, multiple Nash equilibria, including mixed strategy equilibria, can coexist.

The following algorithm gives an approach to use the best response condition to systematically find all Nash equilibrium.

5.2.3 Definition: Support Enumeration Algorithm

The algorithm proceeds as follows:

For a two-player game $(M_r, M_c) \in (\mathbb{R}^{m \times n})^2$, the following algorithm returns all pairs of best responses:

1. For all pairs of supports (subsets of the action space) (I, J):
2. Solve the following equations (to ensure best responses):
   - $\sum_{i \in I} \sigma_{r_i} M_{(c)_{ij}} = v \quad \text{for all } j \in J$
   - $\sum_{j \in J} M_{r_{ij}} \sigma_{(c)_j} = u \quad \text{for all } i \in I$
3. Solve the normalisation and non-negativity constraints:
   - $\sum_{i=1}^{m} \sigma_{r_i} = 1$ and $\sigma_{1_i} \geq 0$ for all i
   - $\sum_{j=1}^{n} \sigma_{c_j} = 1$ and $\sigma_{2_j} \geq 0$ for all j
4. Check the best response condition.

Repeat steps 2–4 for all potential pairs of actions.

Note

The algorithm is described in the context of $N = 2$ players here.

It is a direct application of the best response condition and can be generalised to larger N.

5.2.3.1 Example: Support enumeration algorithm for the coordination game

Let us apply the support enumeration algorithm to the coordination game.

The following supports (subsets of the action space) need to be considered:

$$I \in \{\{r_1\}, \{r_2\}, \{r_1, r_2\}\} \tag{5.6}$$

and

$$J \in \{\{c_1\}, \{c_2\}, \{c_1, c_2\}\} \tag{5.7}$$

For the cases where $| I | = | J | = 1$ steps 2, 3 and 4 of the support enumeration algorithm correspond to finding best responses in the action space. This can be done by highlighting best responses:

$$M_r = \begin{pmatrix} \underline{3} & 1 \\ 0 & \underline{2} \end{pmatrix} \qquad M_c = \begin{pmatrix} \underline{2} & 1 \\ 0 & \underline{3} \end{pmatrix} \tag{5.8}$$

The support enumeration algorithm for $| I | = | J | = 1$ gives the two following Nash equilibria:

$$((1, 0), (1, 0)) \qquad ((0, 1), (0, 1)) \tag{5.9}$$

Note

For $| I | \neq | J |$ we can omit steps 2, 3 and 4 as there is a single given best response to an action in each action space. For example, we do not need to consider specifically if there is a Nash equilibrium when the row player plays the first row and the column player plays both the first and the second column (since the utility for those two columns is not the same. This will be tackled formally in the definition of degenerate games.

The final pair of actions to consider is when $I = (r_1, r_2)$ and $J = (c_1, c_2)$. In this case let:

$$\sigma_1 = (x, 1 - x) \qquad \sigma_2 = (y, 1 - y) \tag{5.10}$$

for $0 < x < 1$ and $0 < y < 1$.

Step 2 corresponds to setting:

$$\begin{aligned}\sum_{i\in I}\sigma_{r_i}M_{(c)_{i1}} &= 2x + 0(1-x) = v\\ \sum_{i\in I}\sigma_{r_i}M_{(c)_{i2}} &= 1x + 3(1-x) = v\end{aligned} \tag{5.11}$$

and

$$\begin{aligned}\sum_{j\in J}M_{r_{1j}}\sigma_{c_j} &= 3y + 1(1-y) = u\\ \sum_{j\in J}M_{r_{2j}}\sigma_{c_j} &= 0y + 2(1-y) = u\end{aligned} \tag{5.12}$$

The particular values of u or v are not required so we can equate these pairs of expressions:

$$\begin{aligned}2x = x + 3 - 3x &\implies x = \frac{3}{4}\\ 3y + 1 - y = 2 - 2y &\implies y = \frac{1}{4}\end{aligned} \tag{5.13}$$

Giving: $\sigma_1 = (3/4, 1/4)$ and $\sigma_2 = (1/4, 3/4)$.

Step 3 already holds as we enforced $\sigma_1 = (x, 1-x)$ and $\sigma_2 = (y, 1-y)$ at the start of our calculations. This is not necessarily the case for larger games.

The final step, step 4, requires us to check the best response condition. This is to check that there exists no better choice of action outside of the chosen set of I and J. This is more relevant for games with larger action spaces but let us nonetheless carry out the calculations:

$$M_r\sigma_2^T = \begin{pmatrix}3/2\\ 3/2\end{pmatrix} \tag{5.14}$$

and

$$\sigma_1 M_c = (3/2 \;\; 3/2) \tag{5.15}$$

as required by the best response condition. The support enumeration algorithm has given 3 Nash equilibria:

$$\left\{((1,0),(1,0)), ((0,1),(0,1)), \left(\left(\frac{3}{4},\frac{1}{4}\right),\left(\frac{1}{4},\frac{3}{4}\right)\right),\right\} \tag{5.16}$$

The support enumeration algorithm is one of many algorithms that can be used to compute Nash equilibrium. Like most of the algorithms it works well for "most" games. When games are "degenerate" it may need more calculations and fail to give all equilibria.

5.2.4 Definition: Degenerate Games

A two player game is called non degenerate if no strategy of support size k has more than k best response actions.

5.2.4.1 Example: Support Enumeration for a degenerate game

Let us use support enumeration for the following game.

$$M_r = \begin{pmatrix}2 & 5\\ 0 & 5\end{pmatrix} \qquad M_c = \begin{pmatrix}2 & 1\\ 0 & 1\end{pmatrix} \tag{5.17}$$

First, we note that this game is degenerate, there are two best responses in action space to the first column:

$$M_r = \begin{pmatrix} \underline{2} & \underline{5} \\ 0 & \underline{5} \end{pmatrix} \qquad M_c = \begin{pmatrix} \underline{2} & 1 \\ 0 & \underline{1} \end{pmatrix} \tag{5.18}$$

Evaluating best responses in action space gives Nash equilibria:

$$((1,0),(1,0)) \qquad ((0,1),(0,1)) \tag{5.19}$$

We need to consider new pairs of supports:

- $\sigma_1 = (x, 1-x)$ and $\sigma_2 = (1,0)$: there is single best response to the first column so nothing else to consider here.
- $\sigma_1 = (x, 1-x)$ and $\sigma_2 = (0,1)$: step 2 holds for all x, step 3 is already satisfied (the vectors are both probability distributions) thus we are left to check the best response condition:

$$\sigma_1 M_c = (2x, 1) \tag{5.20}$$

the only value of x that gives a pair of best response is when the column player has no incentive to move from the support thus $2x = 1 \implies x = 1/2$.

5.3 Exercises

Exercise 5.1:

Use support enumeration to find Nash equilibria for the following games:

1.

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} \tag{5.21}$$

2.

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{5.22}$$

Exercise 5.2:

A soccer player (Player 1) is taking a penalty kick and can shoot either left or right: $S_1 = \{\text{SL}, \text{SR}\}$. The goalie (Player 2) can dive left or right: $S_2 = \{\text{DL}, \text{DR}\}$. The probabilities of scoring a goal (depending on the chosen strategies) are shown in the matrix below:

$$\begin{pmatrix} 0.8 & 0.15 \\ 0.2 & 0.95 \end{pmatrix} \tag{5.23}$$

Assume the utility to Player 1 is the probability of scoring, and the utility to Player 2 is the probability of preventing a goal. What is the Nash equilibrium of this game?

Now suppose Player 1 has a third strategy: shooting in the middle. The new action set becomes $S_1 = \{\text{SL}, \text{SM}, \text{SR}\}$. The updated probability matrix is:

$$\begin{pmatrix} 0.8 & 0.15 \\ 0.5 & 0.5 \\ 0.2 & 0.95 \end{pmatrix} \tag{5.24}$$

Determine the new Nash equilibrium for the extended game.

5.4 Programming

The Nashpy library has an implementation of the support enumeration algorithm. First let us create the payoff matrices and the game:

```
import numpy as np
import nashpy as nash

A = np.array(
    (
        (3, 1),
        (0, 2),
    )
)
B = np.array(
    (
        (2, 1),
        (0, 3),
    )
)

coordination_game = nash.Game(A, B)
print(coordination_game)
```

```
Bi matrix game with payoff matrices:

Row player:
[[3 1]
 [0 2]]

Column player:
[[2 1]
 [0 3]]
```

Now to use the support enumeration algorithm:

```
print(list(coordination_game.support_enumeration()))
```

```
[(array([1., 0.]), array([1., 0.])), (array([0., 1.]), array([0., 1.])),
(array([0.75, 0.25]), array([0.25, 0.75]))]
```

Note

The Nashpy library uses a concept called a generator which allows us to efficiently create the steps of an algorithm without using resources. The call to transform the generator in to a `list` empties the generator to actually go through the steps of the algorithm.

5.5 Notable Research

Support enumeration is as close to a "from first principles" algorithm for computing Nash equilibria. Thus there is no specific paper to point to as per its formulation. Or indeed there is no specific set of

papers that use it for their findings although a lot of papers that consider small games are in essence using support enumeration.

For example in (Chiappori et al., 2002) theoretical results are given regarding penalty kicks that rely on the indifference ensured by the best response condition which is in turn essentially an application of the support enumeration algorithm.

A similar paper applied to another animal conservation is (Lee & Roberts, 2016) in which the authors build a theoretical model of the effectiveness of Rhino horn devaluation. Once again the calculation presented are essentially an application of the support enumeration algorithm.

In (Knight et al., 2017) whilst a different algorithm is used to identify Nash equilibrium for strategic hospital interactions it is somewhat similar to the support enumeration algorithm except that it takes advantage of the specific structure of the game considered.

5.6 Conclusion

This chapter introduced the concept of Nash equilibrium and demonstrated a systematic method for identifying all equilibria in a two-player game using the **support enumeration algorithm**. This method is rooted directly in the best response condition and provides a computational approach that works well for many games, particularly when the number of actions is small.

The support enumeration algorithm is conceptually simple but becomes expensive as action spaces grow; it is most useful for small games.

Table 5.1 summarises the core concepts introduced in this chapter.

Concept	Description
Nash equilibrium	A strategy profile where each player's strategy is a best response to the others'
Support of a strategy	The set of actions played with positive probability
Support enumeration algorithm	Enumerates possible supports and checks conditions for equilibrium
Degenerate game	A game in which some strategy of support size k has more than k best responses

Table 5.1: The main concepts of Nash equilibrium

Attention

Support enumeration embodies the best response condition and provides an accessible, transparent algorithm for computing all equilibria in two-player games.

5.7 Solutions

Solution 5.1: Solution to Exercise 1

We apply the support enumeration algorithm to find all Nash equilibria.

Game 1:

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} \tag{5.25}$$

The row player has 2 actions (r_1, r_2) and the column player has 3 actions (c_1, c_2, c_3).

Pure strategy Nash equilibria.

Best responses for the row player: r_1 is a best response to c_1 $(3 > 2)$ and c_2 $(3 > 1)$; r_2 is a best response to c_3 $(3 > 2)$.

Best responses for the column player: c_3 is a best response to r_1 $(3 > 2 > 1)$; c_2 is a best response to r_2 $(3 > 2 = 2)$.

There are no mutual best response pairs, so no pure strategy Nash equilibrium exists.

Mixed strategy supports.

This game is non-degenerate, so no strategy of support size k has more than k best responses. With $|\ I\ | = 2$, the column player can have at most 2 best responses, so supports with $|\ J\ | = 3$ need not be considered.

$I = \{r_1, r_2\},\ J = \{c_1, c_2\}$: Let $\sigma_c = (y_1, y_2, 0)$ with $y_1 + y_2 = 1$. Row player indifference requires $3y_1 + 3y_2 = 2y_1 + y_2$, giving $3 = y_1 + 1$, so $y_1 = 2$. Invalid.

$I = \{r_1, r_2\}$, $J = \{c_1, c_3\}$: Let $\sigma_c = (y_1, 0, y_3)$ with $y_1 + y_3 = 1$.

Row player indifference: $3y_1 + 2y_3 = 2y_1 + 3y_3 \implies y_1 = y_3 = \frac{1}{2}$.

Column player indifference: $2x + 2(1 - x) = 3x + 2(1 - x) \implies 2 = x + 2 \implies x = 0$. Contradicts support $\{r_1, r_2\}$. No NE.

$I = \{r_1, r_2\}$, $J = \{c_2, c_3\}$: Let $\sigma_c = (0, y_2, y_3)$ with $y_2 + y_3 = 1$.

Row player indifference: $3y_2 + 2y_3 = y_2 + 3y_3 \implies 2y_2 = y_3$.

With $y_2 + y_3 = 1$: $y_2 = \frac{1}{3}$, $y_3 = \frac{2}{3}$.

Column player indifference: $x + 3(1 - x) = 3x + 2(1 - x) \implies 3 - 2x = x + 2 \implies x = \frac{1}{3}$.

Both x and (y_2, y_3) are valid probability distributions. Checking the best response condition:

$$A\sigma_c^T = \begin{pmatrix} 3 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1/3 \\ 2/3 \end{pmatrix} = \begin{pmatrix} 7/3 \\ 7/3 \end{pmatrix} \tag{5.26}$$

$$\sigma_r B = (1/3 \ \ 2/3) \begin{pmatrix} 2 & 1 & 3 \\ 2 & 3 & 2 \end{pmatrix} = (2 \ \ 7/3 \ \ 7/3) \tag{5.27}$$

Both rows give equal payoff $7/3$, and column c_1 (outside the support) gives $2 < 7/3$, so the best response condition is satisfied.

Nash equilibria for Game 1:

$$\left\{ \left(\left(\frac{1}{3}, \frac{2}{3} \right),\ \left(0, \frac{1}{3}, \frac{2}{3} \right) \right) \right\} \tag{5.28}$$

Here is some code to confirm the calculation:

```
import nashpy as nash
import numpy as np
```

```
A = np.array([[3, 3, 2], [2, 1, 3]])
B = np.array([[2, 1, 3], [2, 3, 2]])
game_1 = nash.Game(A, B)
for eq in game_1.support_enumeration():
    print(eq)
```

```
(array([0.33333333, 0.66666667]), array([0.        , 0.33333333, 0.66666667]))
```

Game 2:

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{5.29}$$

Pure strategy Nash equilibria.

Row player: r_1 is a best response to c_1 ($3 > 2$); r_2 is a best response to c_2 ($7 > -1$). Column player: c_2 is a best response to r_1 ($1 > -3$); c_1 is a best response to r_2 ($1 > -6$). There are no mutual best response pairs, so no pure NE exists.

Mixed strategy with $I = \{r_1, r_2\}$, $J = \{c_1, c_2\}$:

Let $\sigma_r = (x, 1-x)$ and $\sigma_c = (y, 1-y)$.

Row player indifference:

$$3y - (1-y) = 2y + 7(1-y) \implies 4y - 1 = 7 - 5y \implies y = \frac{8}{9} \tag{5.30}$$

Column player indifference:

$$-3x + (1-x) = x - 6(1-x) \implies 1 - 4x = 7x - 6 \implies x = \frac{7}{11} \tag{5.31}$$

Both values lie in $(0, 1)$. Checking the best response condition:

$$A\sigma_c^T = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix} \begin{pmatrix} 8/9 \\ 1/9 \end{pmatrix} = \begin{pmatrix} 23/9 \\ 23/9 \end{pmatrix} \tag{5.32}$$

$$\sigma_r B = \begin{pmatrix} 7/11 & 4/11 \end{pmatrix} \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} = \begin{pmatrix} -17/11 & -17/11 \end{pmatrix} \tag{5.33}$$

Both rows and both columns yield equal payoffs, confirming the equilibrium.

Nash equilibria for Game 2:

$$\left\{ \left(\left(\frac{7}{11}, \frac{4}{11} \right), \left(\frac{8}{9}, \frac{1}{9} \right) \right) \right\} \tag{5.34}$$

Here is some code to confirm the calculations:

```
A = np.array([[3, -1], [2, 7]])
B = np.array([[-3, 1], [1, -6]])
game_2 = nash.Game(A, B)
for eq in game_2.support_enumeration():
    print(eq)
```

```
(array([0.63636364, 0.36363636]), array([0.88888889, 0.11111111]))
```

Solution 5.2: Solution to Exercise 2

The payoff matrices are:

$$M_r = \begin{pmatrix} 0.8 & 0.15 \\ 0.2 & 0.95 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0.2 & 0.85 \\ 0.8 & 0.05 \end{pmatrix} \tag{5.35}$$

where $M_c = 1 - M_r$ since the goalkeeper's utility is the probability of saving.

Pure strategy best responses.

Row player: SL is a best response to DL ($0.8 > 0.2$); SR is a best response to DR ($0.95 > 0.15$). Column player: DR is a best response to SL ($0.85 > 0.20$); DL is a best response to SR ($0.8 > 0.05$). There are no mutual best response pairs, so no pure NE exists.

Mixed strategy NE with $I = \{\text{SL}, \text{SR}\}$, $J = \{\text{DL}, \text{DR}\}$:

Let $\sigma_1 = (x, 1-x)$ and $\sigma_2 = (y, 1-y)$.

Kicker indifference:

$$0.8y + 0.15(1-y) = 0.2y + 0.95(1-y) \implies 1.4y = 0.8 \implies y = \frac{4}{7} \tag{5.36}$$

Goalkeeper indifference:

$$0.2x + 0.8(1-x) = 0.85x + 0.05(1-x) \implies 0.75 = 1.4x \implies x = \frac{15}{28} \tag{5.37}$$

Both values lie in $(0, 1)$. The Nash equilibrium of the two-action game is:

$$\sigma_1^* = \left(\frac{15}{28}, \frac{13}{28}\right) \qquad \sigma_2^* = \left(\frac{4}{7}, \frac{3}{7}\right) \tag{5.38}$$

```python
import nashpy as nash
import numpy as np

M_r = np.array([[0.8, 0.15], [0.2, 0.95]])
M_c = 1 - M_r
two_action_game = nash.Game(M_r, M_c)
for eq in two_action_game.support_enumeration():
    print(eq)
```

```
(array([0.53571429, 0.46428571]), array([0.57142857, 0.42857143]))
```

Extended game with $S_1 = \{\text{SL}, \text{SM}, \text{SR}\}$:

$$M_r = \begin{pmatrix} 0.8 & 0.15 \\ 0.5 & 0.5 \\ 0.2 & 0.95 \end{pmatrix} \qquad M_c = \begin{pmatrix} 0.2 & 0.85 \\ 0.5 & 0.5 \\ 0.8 & 0.05 \end{pmatrix} \tag{5.39}$$

There is no pure NE by the same cycling argument as before. We check each possible kicker support against $J = \{\text{DL}, \text{DR}\}$.

Support $\{\text{SL}, \text{SR}\}$: The equilibrium $\sigma_2^* = (4/7, 3/7)$ from the two-action game remains a candidate. We verify SM is not a profitable deviation for the kicker:

$$u_r(\text{SM},\ \sigma_2^*) = 0.5 \cdot \frac{4}{7} + 0.5 \cdot \frac{3}{7} = 0.5 \tag{5.40}$$

$$u_r(\text{SL},\ \sigma_2^*) = 0.8 \cdot \frac{4}{7} + 0.15 \cdot \frac{3}{7} = \frac{3.65}{7} \approx 0.521 \tag{5.41}$$

Since $0.5 < 0.521$, SM is not a best response, so $\sigma_1^* = (15/28,\ 0,\ 13/28)$ and $\sigma_2^* = (4/7,\ 3/7)$ is a valid NE.

Support $\{\text{SL}, \text{SM}\}$: Goalkeeper indifference requires $0.2\, x_{\text{SL}} + 0.5\, x_{\text{SM}} = 0.85\, x_{\text{SL}} + 0.5\, x_{\text{SM}}$, which gives $x_{\text{SL}} = 0$. Contradicts the support. No NE.

Support $\{\text{SM}, \text{SR}\}$: Goalkeeper indifference requires $x_{\text{SR}} = 0$. Contradicts the support. No NE.

Support $\{\text{SL}, \text{SM}, \text{SR}\}$: Kicker indifference requires $\text{SL} = \text{SM} = \text{SR}$ under $\sigma_2 = (y, 1-y)$. From $\text{SL} = \text{SR}$ we get $y = 4/7$, but from $\text{SL} = \text{SM}$ we get $y = 7/13 \neq 4/7$. No full-support NE exists.

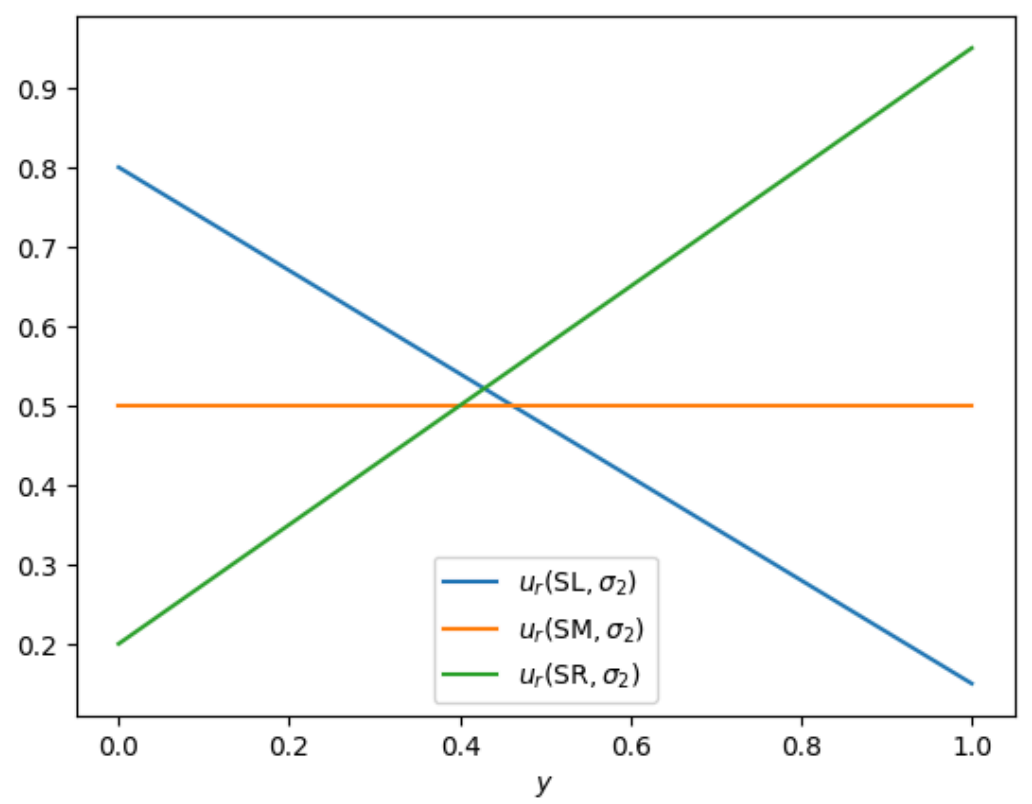
Conclusion: The unique Nash equilibrium in the extended game is:

$$\sigma_1^* = \left(\frac{15}{28},\ 0,\ \frac{13}{28}\right), \qquad \sigma_2^* = \left(\frac{4}{7},\ \frac{3}{7}\right) \tag{5.42}$$

The middle shot is never played at equilibrium because its scoring probabilities are dominated by the mixing of SL and SR, giving the kicker no strategic advantage from including it. This can been seen numerically by plotting the utilities as a function of the $\sigma_2 = (y, 1-y)$. At no value of y is the middle shot ever a best response.

```
import matplotlib.pyplot as plt

M_r = np.array([[0.8, 0.15], [0.5, 0.5], [0.2, 0.95]])

plt.plot(M_r[0], label=r"$u_r(\text{SL}, \sigma_2)$")
plt.plot(M_r[1], label=r"$u_r(\text{SM}, \sigma_2)$")
plt.plot(M_r[2], label=r"$u_r(\text{SR}, \sigma_2)$")
plt.xlabel("$y$")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f686d25dfd0>
```



Let us confirm the equilibria:

```
M_c = 1 - M_r
three_action_game = nash.Game(M_r, M_c)
for eq in three_action_game.support_enumeration():
    print(eq)
```

```
(array([0.53571429, 0.        , 0.46428571]), array([0.57142857, 0.42857143]))
```

6. Subgame Perfection

Nash equilibrium can be sustained by off-path promises that would never actually be honoured. This chapter introduces subgame perfect equilibrium, which rules out such implausible promises by requiring rationality at every point in the game tree, not only along the equilibrium path.

6.1 Motivating Example

We will motivate this chapter using a special case of Selten's Chain Store Paradox (Selten, 1978) which models an incumbent chain of stores that is well established and an entrant to the market.

Campus Caffeine, a mobile coffee van, is considering parking near a university where Latte Lords, a well-established bricks-and-mortar café, has been operating unchallenged. If Campus Caffeine **enters**, Latte Lords can slash prices for students (**Fight**) or maintain their premium pricing (**Accommodate**). If Campus Caffeine stays out (**withdraws**), Latte Lords keeps enjoying high margins.

There are three outcomes which are shown as an extensive form game in Figure 6.1:

- Campus Caffeine chooses to withdraw and does not enter the market: they receive a utility of 1 and Latte Lords continues to have a monopoly with which corresponds to a utility of 5.
- Campus Caffeine enters the market and Latte Lords competes: this leads to low prices that give neither company a profit. They both obtain a utility of 0.
- Campus Caffeine enters and Latte Lords accommodates: they both share the market obtaining a utility of 2.

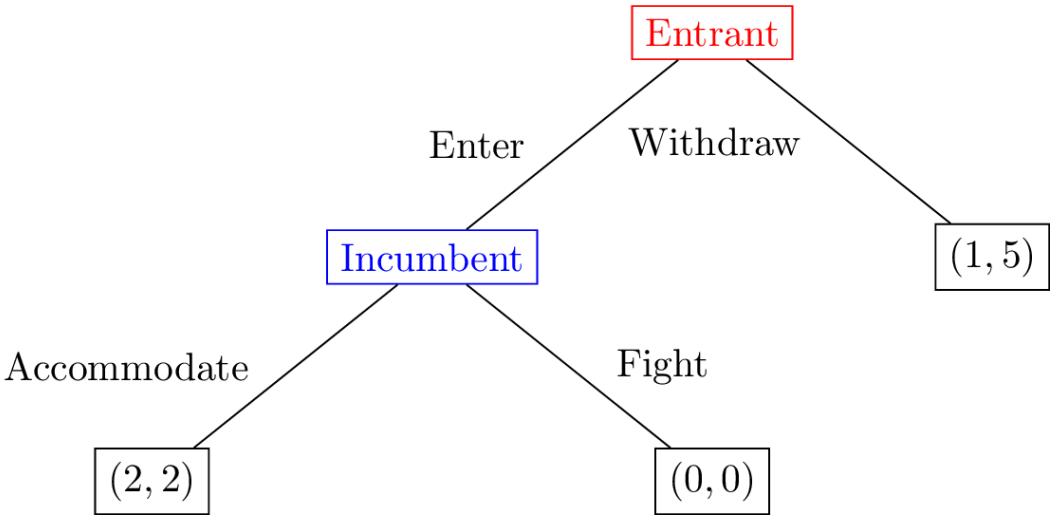


Figure 6.1: The extensive form game between Campus Caffeine and Latte Lords.

As described in the Mapping Extensive Form Games to Normal Form section it is possible to map an extensive form game to a normal form game. In this case we have:

$$\mathcal{A}_1 = \{\text{Accommodate}, \text{Fight}\} \mathcal{A}_2 = \{\text{Enter}, \text{Withdraw}\} \tag{6.1}$$

giving:

$$M_1 = \begin{pmatrix} 2 & 5 \\ 0 & 5 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \tag{6.2}$$

The incumbent can threaten to fight the entrant however the entrant knows that this threat if carried out would be harmful to both of them. Thus the threat is in essence an empty threat: the entrant will enter and the incumbent will use their best response which is to accommodate.

This is an example of a degenerate game, the Nash equilibria was considered in Example: Support Enumeration for a degenerate game showing that there are in fact 3 of them:

- $((1,0),(1,0))$: the entrant enters the market and the incumbent accommodates.
- $((0,1),(0,1))$: the entrant withdraws and the incumbent fights.
- $\left(\left(\frac{1}{2},\frac{1}{2}\right),(0,1)\right)$: the incumbent mixes equally between accommodating and fighting while the entrant withdraws.

This chapter will give the mathematical vocabulary to describe the difference between these equilibria.

6.2 Theory

To identify emergent behaviour in extensive form games we assume that players not only attempt to optimise their overall utility but optimise their utility conditional on any information set.

6.2.1 Definition: Sequential rationality

Sequential rationality: An optimal strategy for a player should maximise that player's expected payoff, conditional on every information set at which that player has a decision.

With this notion in mind we can now define an analysis technique for extensive form games:

6.2.2 Definition: Backward induction

Backward induction: This is the process of analysing a game from back to front. At each information set we remove strategies that are dominated.

6.2.2.1 Example

Let us consider the game shown in Figure 6.2

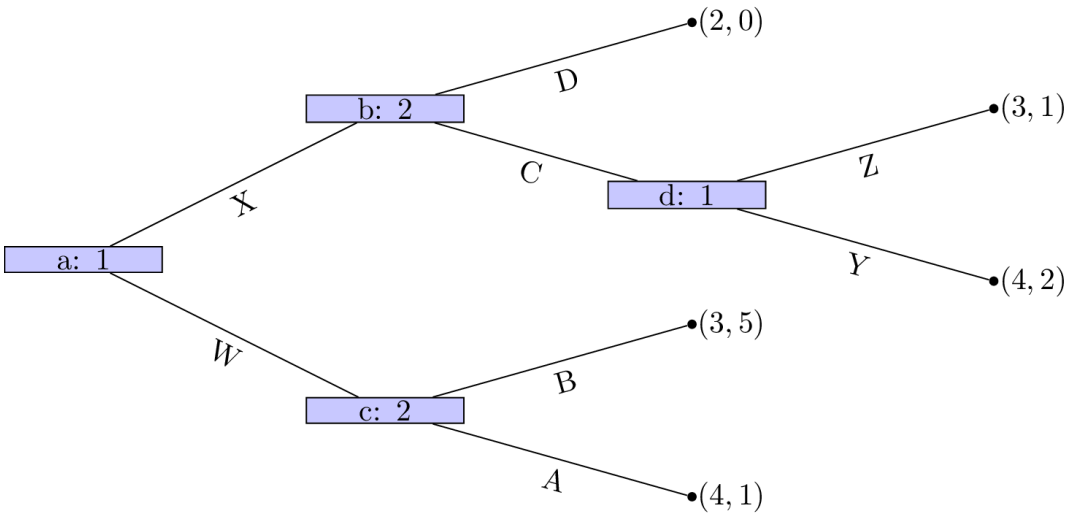


Figure 6.2: An extensive form game that we will use backwards induction on.

We see that at node (d) that Z is a dominated action. So the game reduces as shown in Figure 6.3.

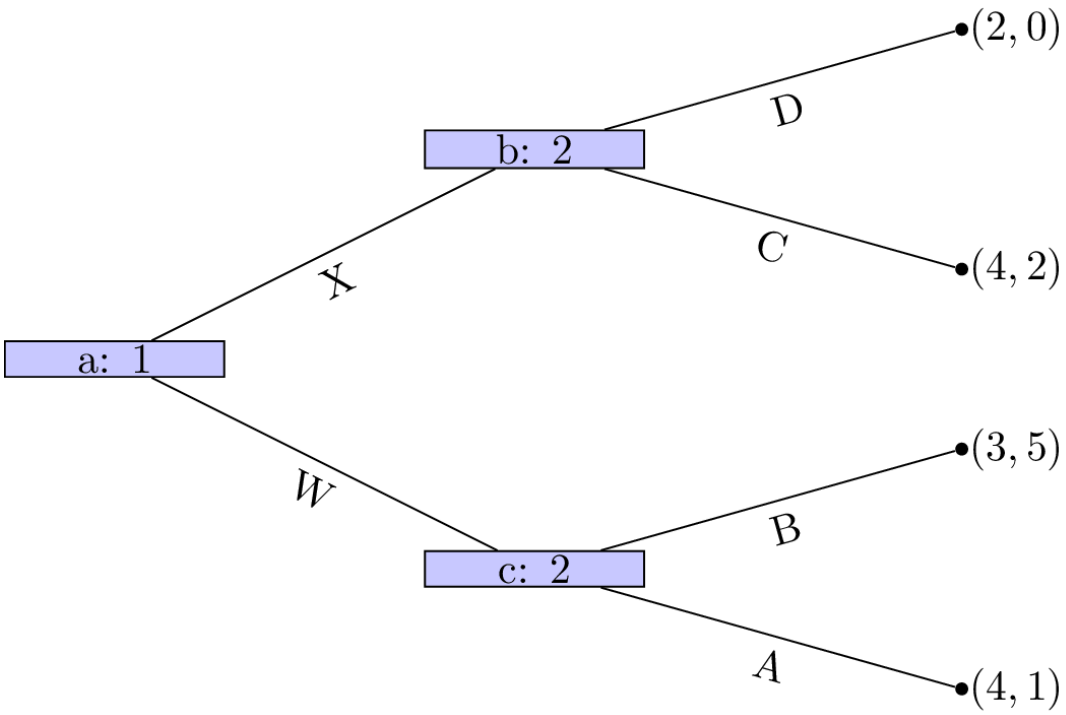


Figure 6.3: Reducing the game because Z is a dominated action.

Player 1s strategy profile is (Y). At node (c) A is a dominated action so that the game reduces as shown in Figure 6.4.

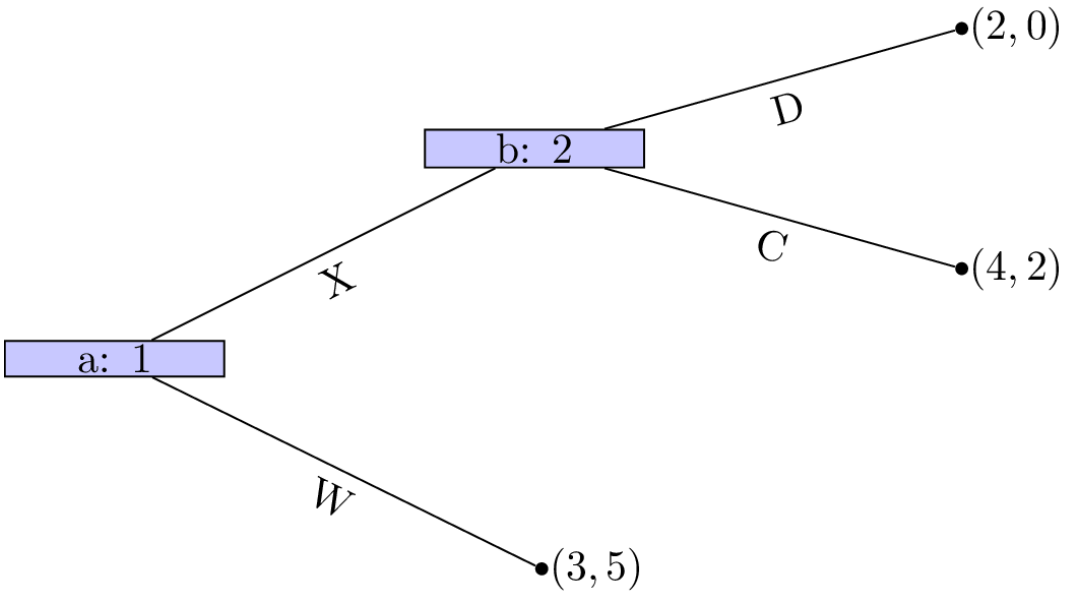


Figure 6.4: Reducing the game because A is a dominated action.

Player 2s strategy profile is (B). At node (b) D is a dominated action so that the game reduces as shown.

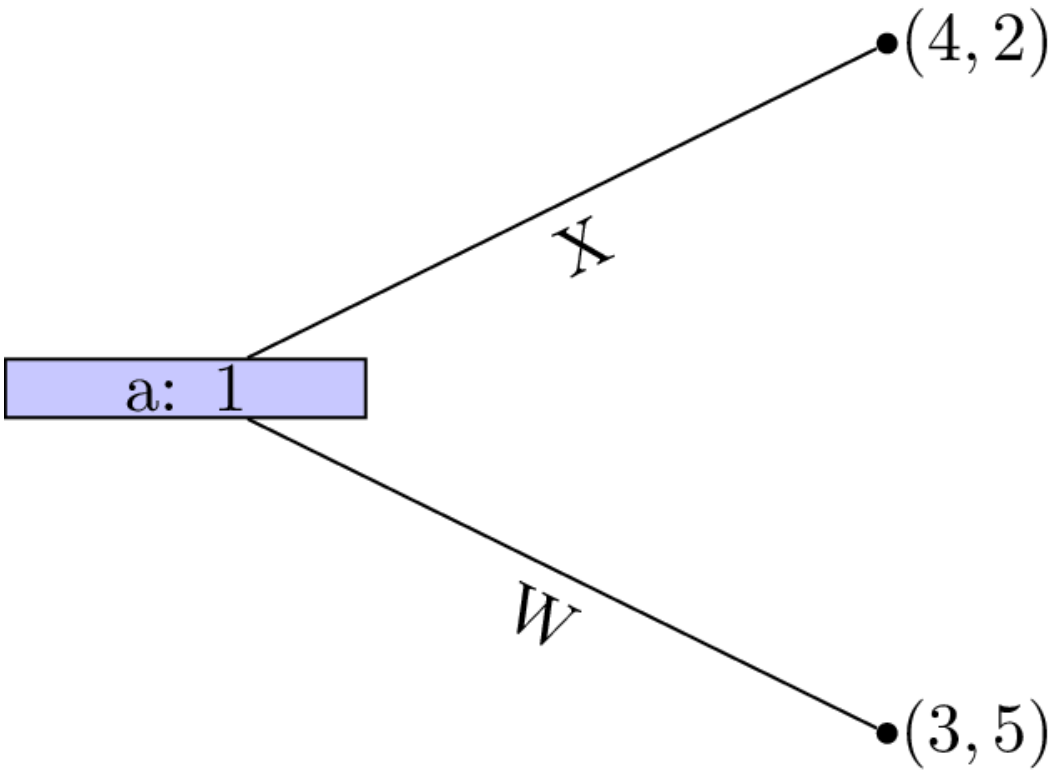


Figure 6.5: Reducing the game because D is a dominated action.

Player 2s strategy profile is thus (C,B) and finally strategy W is dominated for player 1 whose strategy profile is (X,Y). This pair of strategies form a Nash equilibrium.

6.2.3 Theorem of existence of Nash equilibrium in games of perfect information.

Every finite game with perfect information has a Nash equilibrium in pure strategies. Backward induction identifies an equilibrium.

Proof:

We argue by induction on the number of nodes of the game tree. If the game has a single node it is a terminal node, the payoffs are already assigned, and there is nothing to choose. Otherwise let x be the root, at which some player i acts. Each action of i at x leads to a subtree, and because the game has perfect information each subtree is itself a finite perfect-information game with strictly fewer nodes. By the induction hypothesis each subtree has a Nash equilibrium in pure strategies found by backward induction; fix one, and let $u_i(a)$ be the payoff it awards player i after action a. Player i chooses an action $a^\star$ maximising $u_i(a)$ (ties are broken arbitrarily, each choice giving a valid equilibrium), and every other player plays their prescribed subtree strategy.

This defines a pure strategy profile for the whole game. To see that it is a Nash equilibrium, note that by sequential rationality no player can gain by deviating within any subtree, since the profile restricts to an equilibrium of that subtree; and player i cannot gain at the root, since $a^\star$ maximises their payoff given the continuation. As no single player has a profitable deviation, the profile is a Nash equilibrium, and by construction it is obtained by backward induction.

6.2.4 Definition: Subgame

In an extensive form game, a node x is said to **initiate a subgame** if and only if x and all successors of x are in information sets containing only successors of x.

6.2.4.1 Example: a game where all nodes initiate subgames

A game where all nodes initiate a subgame is given in Figure 6.6.

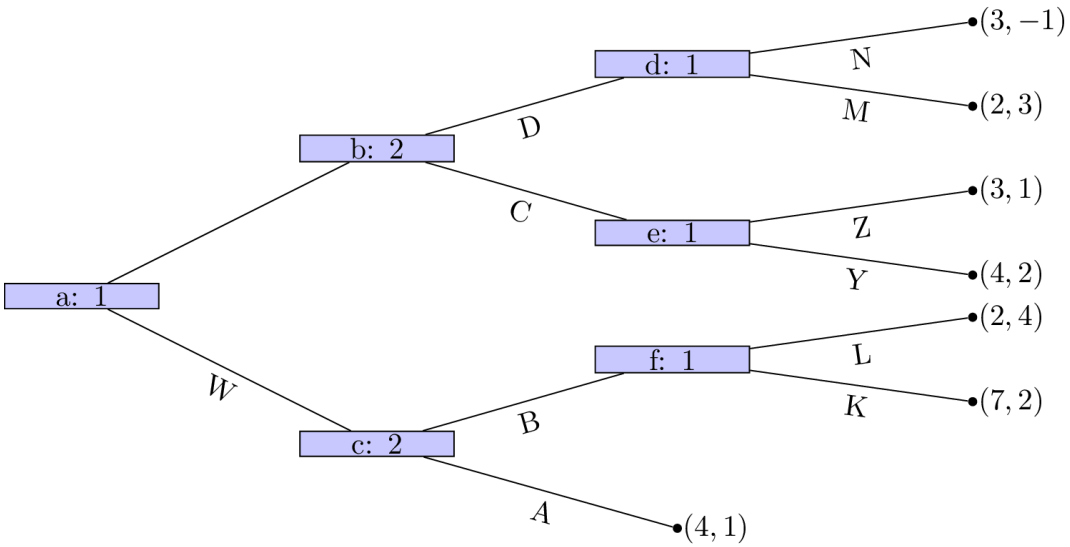


Figure 6.6: An extensive form game where all nodes initiate a subgame.

6.2.4.2 Example: a game where all nodes do not initiate subgames

A game **that does not have perfect information** nodes c, f and b initiate subgames but both of b's successors do not is shown in Figure 6.7

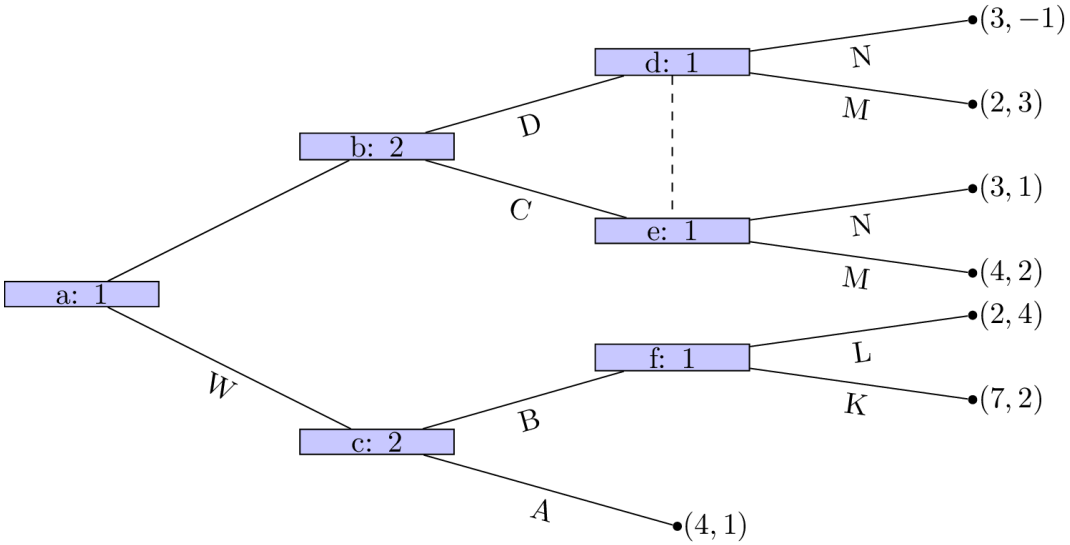


Figure 6.7: An extensive form game where all nodes do not initiate a subgame.

The notion of a subgame leads us to define a specific property of some Nash equilibrium.

6.2.5 Definition: Subgame perfect equilibrium

A subgame perfect Nash equilibrium is a Nash equilibrium in which the strategy profiles specify Nash equilibria for every subgame of the game.

Note

This includes subgames that might not be reached during play.

Let us consider the example in Figure 6.8.

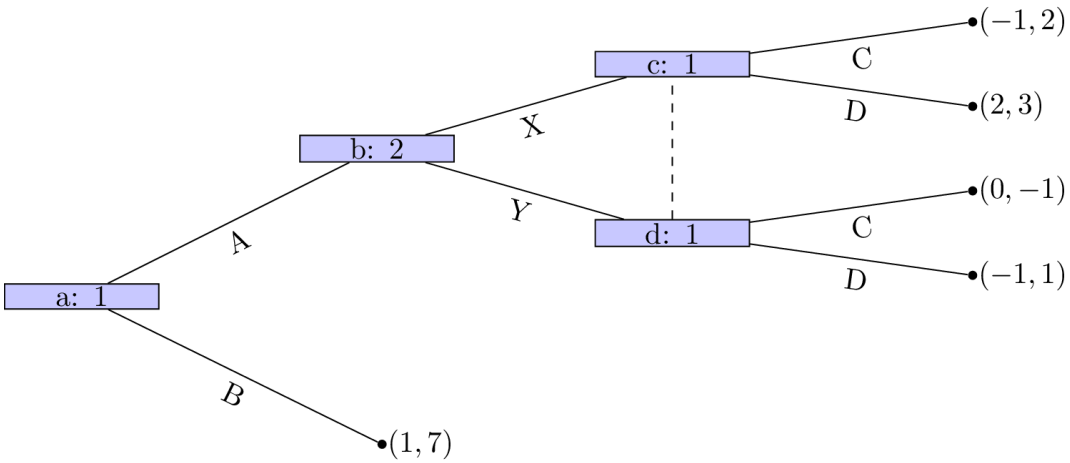


Figure 6.8: An game with subgame perfect equilibrium.

Let us build the corresponding normal form game:

$$A_1 = \{AC, AD, BC, BD\} \tag{6.3}$$

and

$$A_2 = \{X, Y\} \tag{6.4}$$

using the above ordering we have:

$$M_1 = \begin{pmatrix} -1 & 0 \\ 2 & -1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 7 & 7 \\ 7 & 7 \end{pmatrix} \tag{6.5}$$

The Nash equilibria for the above game (found by inspecting best responses in action space) are:

$$\{(AD, X), (BC, Y), (BD, Y)\} \tag{6.6}$$

If we take a look at the normal form game representation of the subgame initiated at node b with action sets:

$$A_1 = \{C, D\} \text{ and } A_2 = \{X, Y\} \tag{6.7}$$

we have:

$$M_1 = \begin{pmatrix} -1 & 0 \\ 2 & -1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} \tag{6.8}$$

We see that the (unique) Nash equilibria for the above game is (D, X). Thus the only subgame perfect equilibria of the *entire* game is $\{AD, X\}$.

Note

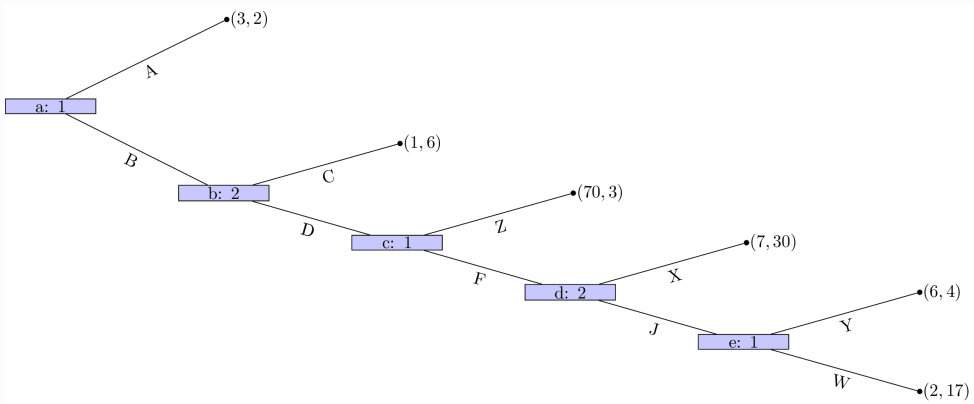
In games with perfect information, the Nash equilibrium obtained through backwards induction is subgame perfect.

6.3 Exercises

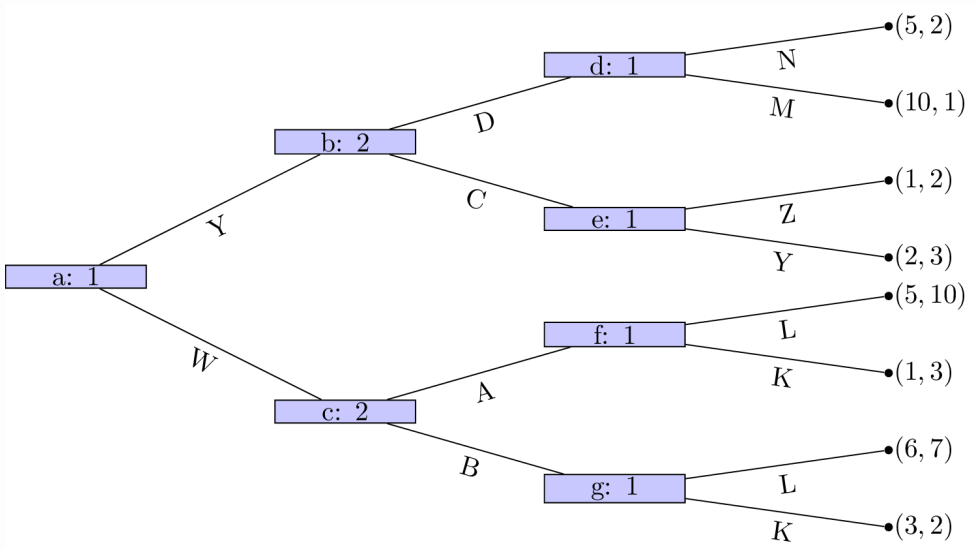
Exercise 6.1:

Obtain the Nash equilibrium for the following games using backward induction:

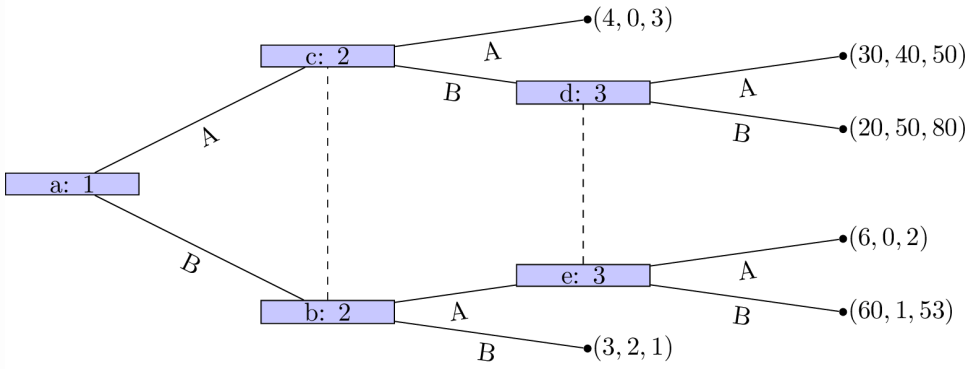
1.



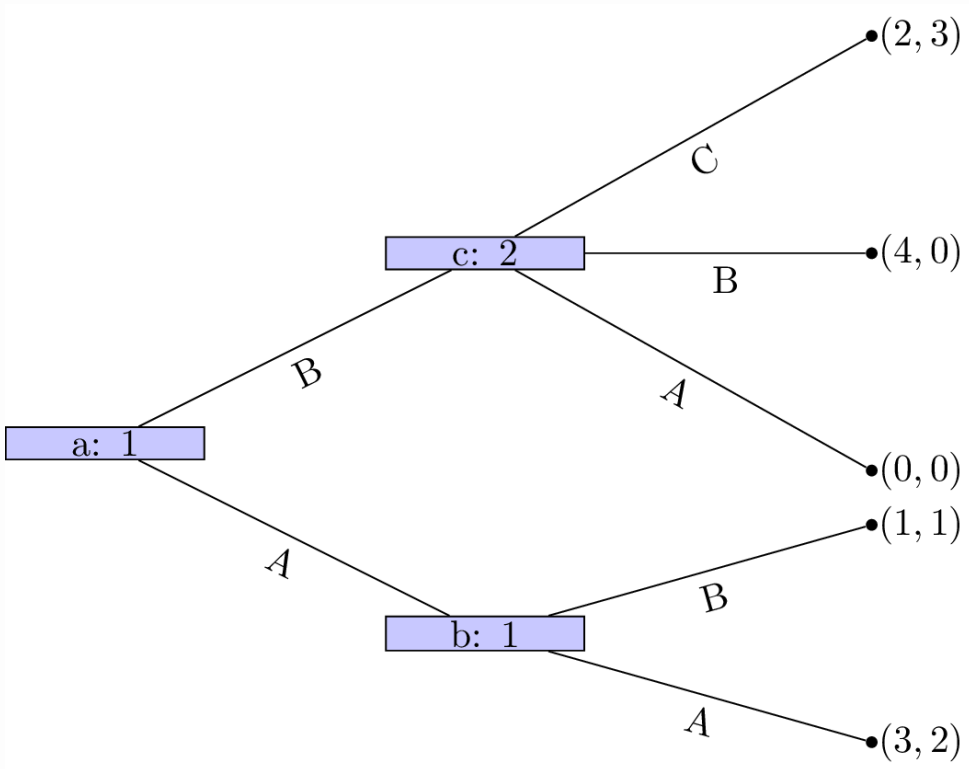
2.



3.



4.



Exercise 6.2:

Player 1 chooses a number $x \geq 0$, which Player 2 observes. Then, both players simultaneously and independently choose real numbers $y_1, y_2 \in \mathbb{R}$. The utility functions are:

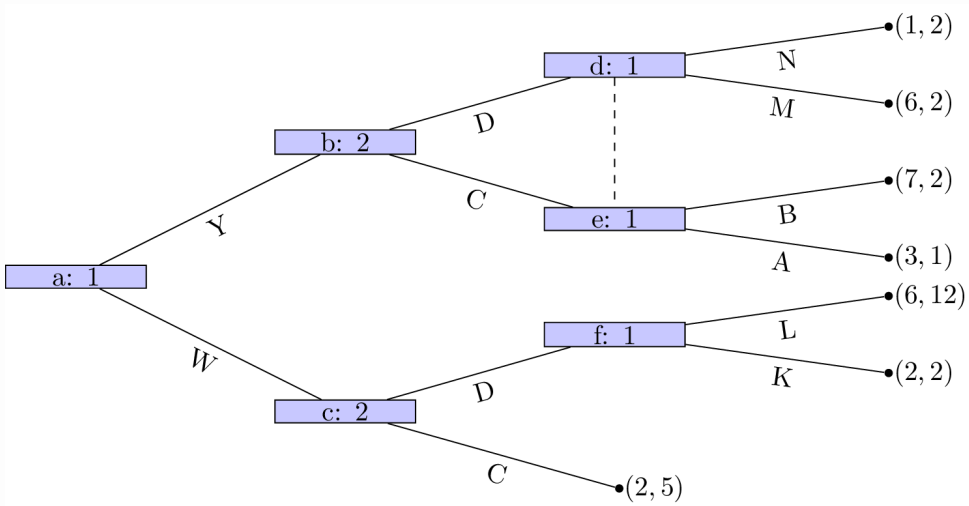
- Player 1: $2y_2y_1 + xy_1 - y_1^2 - \frac{x^3}{3}$
- Player 2: $-(y_1 - 2y_2)^2$

Find the subgame perfect equilibrium of this game.

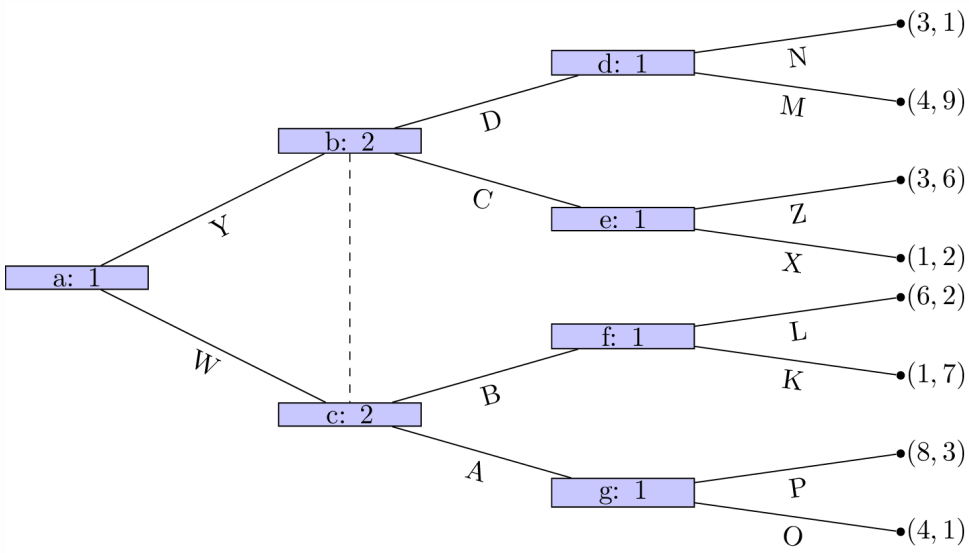
Exercise 6.3:

For each of the following extensive form games:

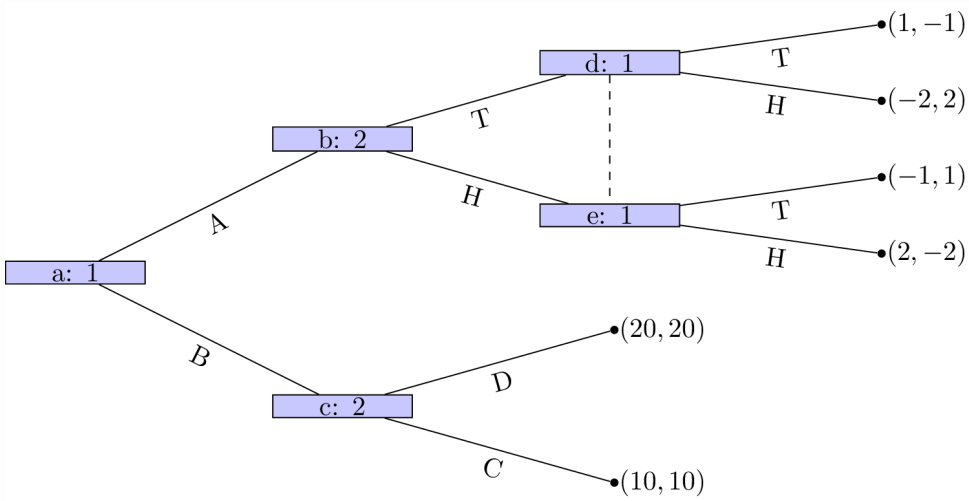
1.



2.



3.



- Identify all subgames.
- Derive the corresponding normal form representations.
- Find all Nash equilibria.
- Identify which are subgame perfect.

Exercise 6.4:

Two ice cream sellers choose locations along a beach represented by $[0, 1]$. Customers are uniformly distributed and always go to the nearest seller.

- **Player 1** chooses $x_1 \in [0, 1]$
- **Player 2** observes x_1, then chooses $x_2 \in [0, 1]$

Each seller's payoff is the proportion of customers they serve. Assume:

- If $x_1 = x_2$, each serves 50%.
- If $x_1 < x_2$, Player 1 serves $\left[0, \frac{x_1+x_2}{2}\right]$, and Player 2 the rest.
- If $x_2 < x_1$, the roles reverse.

1. Write the payoff functions of each player.
2. Derive Player 2's best response function $x_2^*(x_1)$.
3. Use backward induction to find the subgame perfect equilibrium.

Hint: Think geometrically about the midpoint between locations.

6.4 Programming

6.4.1 Using Gambit to study an extensive form game

The `pygambit` library can compute equilibria of extensive form games. We define Selten's Chain Store Paradox as follows:

```
import pygambit as gbt

g = gbt.Game.new_tree(players=["Incumbent", "Entrant"],
                      title="1 stage Selten's Chain Store Paradox")
g.append_move(g.root, "Entrant", ["Enter", "Withdraw"])
g.append_move(g.root.children[0], "Incumbent", ["Accomodate", "Fight"])
g.set_outcome(g.root.children[1], g.add_outcome([5, 1], label="No entry"))
g.set_outcome(g.root.children[0].children[0], g.add_outcome([2, 2], label="Shared
Market"))
g.set_outcome(g.root.children[0].children[1], g.add_outcome([0, 0],
label="Competition"))
print(g)
```

```
Game(title='1 stage Selten's Chain Store Paradox')
```

This creates the extensive form game. To convert it to a normal form:

```
print(f"Normal form payoff matrices: {g.to_arrays(dtype=float)}")
```

```
Normal form payoff matrices: [array([[2.0, 5.0],
       [0.0, 5.0]], dtype=object), array([[2.0, 1.0],
       [0.0, 1.0]], dtype=object)]
```

Note

We use `dtype=float` so that results are returned in floating-point instead of exact rational arithmetic.

To compute the pure-strategy Nash equilibria:

```
result = gbt.nash.enumpure_solve(g)
print(f"Pure-strategy Nash equilibria: {result.equilibria}")
```

```
Pure-strategy Nash equilibria: [[[Rational(1, 1), Rational(0, 1)], [Rational(1,
1), Rational(0, 1)]], [[Rational(0, 1), Rational(1, 1)], [Rational(0, 1),
Rational(1, 1)]]]
```

6.5 Notable Research

The concept of subgame perfection was first introduced in (Selten, 1965) and later reformulated in English in (Selten & Bielefeld, 1988), both by Reinhard Selten. The motivating example for this chapter, the Chain Store Paradox, appears in (Selten, 1978), where Selten considers multiple sequential market entrants and examines the credibility of entry-deterrence strategies.

The framework was extended in (Milgrom & Roberts, 1982), which introduced **asymmetric information** and **reputation effects**. These additions showed how players might sustain seemingly irrational threats (such as fighting entry) by considering the beliefs and inferences of future opponents.

The idea of subgame perfection was further refined in (Kreps & Wilson, 1982), which introduced the concept of **sequential equilibrium**, and in (Grossman & Perry, 1986), which developed **perfect sequential equilibrium**. These refinements allow for the analysis of games where off-equilibrium beliefs matter, providing stronger predictions in extensive form games with imperfect information.

6.6 Conclusion

Subgame perfection refines the concept of Nash equilibrium by requiring that players' strategies form a Nash equilibrium not just in the game as a whole, but in every subgame, including those off the equilibrium path. This refinement eliminates non-credible threats and ensures sequential rationality at every decision point. Table Table 6.1 summaries the main concepts of this chapter.

Concept	Description
Sequential rationality	Players optimise at each information set
Backward induction	Method to compute subgame perfect equilibria in perfect information
Subgame	Portion of a game starting at a decision node and fully self-contained
Subgame perfect equilibrium	A strategy profile that is a Nash equilibrium in every subgame

Table 6.1: The main concepts for Subgame Perfectoin

We illustrated these ideas with Selten's Chain Store Paradox, which highlights how subgame perfection rules out the incumbent's non-credible threat to fight.

Attention

In games with perfect information, backward induction not only gives a Nash equilibrium; it guarantees a subgame perfect one.

6.7 Solutions

Solution 6.1: Solution to Exercise 1

Game 1.

Starting at the deepest node, node e (Player 1): Y gives payoff $6 > 2$, so Player 1 chooses Y, replacing node e with $(6, 4)$.

At node d (Player 2): X gives payoff $30 > 4$, so Player 2 chooses X, replacing node d with $(7, 30)$.

At node c (Player 1): Z gives payoff $70 > 7$, so Player 1 chooses Z, replacing node c with $(70, 3)$.

At node b (Player 2): C gives payoff $6 > 3$, so Player 2 chooses C, replacing node b with $(1, 6)$.

At node a (Player 1): A gives payoff $3 > 1$, so Player 1 chooses A.

The Nash equilibrium is $(\boldsymbol{AZY}, \boldsymbol{CX})$ with outcome $(3, 2)$.

Game 2.

At node d (Player 1): M gives $10 > 5$, so Player 1 chooses M, replacing node d with $(10, 1)$.

At node e (Player 1): Y gives $2 > 1$, so Player 1 chooses Y, replacing node e with $(2, 3)$.

At node f (Player 1): L gives $5 > 1$, so Player 1 chooses L, replacing node f with $(5, 10)$.

At node g (Player 1): L gives $6 > 3$, so Player 1 chooses L, replacing node g with $(6, 7)$.

At node b (Player 2): C gives $3 > 1$, so Player 2 chooses C, replacing node b with $(2, 3)$.

At node c (Player 2): A gives $10 > 7$, so Player 2 chooses A, replacing node c with $(5, 10)$.

At node a (Player 1): W gives $5 > 2$, so Player 1 chooses W.

The Nash equilibrium is $(\boldsymbol{WMYLL}, \boldsymbol{AC})$ with outcome $(5, 10)$.

Game 3.

This game has imperfect information: Player 2's nodes b and c form one information set, and Player 3's nodes d and e form another. Backward induction is applied by identifying dominant actions at each information set.

At Player 3's information set $\{d, e\}$: at node d, action B gives $80 > 50$; at node e, action B gives $53 > 2$. So B dominates A for Player 3 at both nodes.

With Player 3 playing B, node d is replaced by $(20, 50, 80)$ and node e by $(60, 1, 53)$.

At Player 2's information set $\{b, c\}$: at node b, action B gives $2 > 1$ (leading to $(3, 2, 1)$ vs $(60, 1, 53)$); at node c, action B gives $50 > 0$ (leading to $(20, 50, 80)$ vs $(4, 0, 3)$). So B dominates A for Player 2 at both nodes.

With Player 2 playing B: node b is replaced by $(3, 2, 1)$ and node c by $(20, 50, 80)$.

At node a (Player 1): A gives $20 > 3$, so Player 1 chooses A.

The Nash equilibrium is $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{B})$ with outcome $(20, 50, 80)$.

Game 4.

At node b (Player 1): A gives $3 > 1$, so Player 1 chooses A, replacing node b with $(3, 2)$.

At node c (Player 2): C gives $3 > 0 = 0$, so Player 2 chooses C, replacing node c with $(2, 3)$.

At node a (Player 1): A gives $3 > 2$, so Player 1 chooses A.

The Nash equilibrium is $(\boldsymbol{AA}, \boldsymbol{C})$ with outcome $(3, 2)$.

Solution 6.2: Solution to Exercise 2

We solve by backward induction. Player 1 moves first (choosing $x \geq 0$), then both players simultaneously choose $y_1, y_2 \in \mathbb{R}$.

Step 1: Solve the simultaneous subgame for fixed x.

Player 2 maximises $-(y_1 - 2y_2)^2$, which requires $y_1 - 2y_2 = 0$, giving:

$$\tilde{y}_2(y_1) = \frac{y_1}{2} \tag{6.9}$$

Player 1 maximises $2y_2y_1 + xy_1 - y_1^2 - x^3/3$ over y_1. The first-order condition is:

$$\frac{\partial u_1}{\partial y_1} = 2y_2 + x - 2y_1 = 0 \Longrightarrow \tilde{y}_1(y_2) = \frac{2y_2 + x}{2} \tag{6.10}$$

Solving the two best responses simultaneously by substituting $\tilde{y}_2 = y_1/2$ into $\tilde{y}_1$:

$$y_1 = \frac{2(y_1/2) + x}{2} = \frac{y_1 + x}{2} \Longrightarrow y_1 = x, \qquad y_2 = \frac{x}{2} \tag{6.11}$$

Step 2: Optimise Player 1's choice of x.

Substituting $y_1 = x$ and $y_2 = x/2$ into u_1:

$$u_1 = 2 \cdot \frac{x}{2} \cdot x + x \cdot x - x^2 - \frac{x^3}{3} = x^2 - \frac{x^3}{3} \tag{6.12}$$

The first-order condition $\frac{du_1}{dx} = 2x - x^2 = x(2 - x) = 0$ gives $x = 0$ or $x = 2$. The second-order condition $\frac{d^2u_1}{dx^2} = 2 - 2x$ equals $-2 < 0$ at $x = 2$ (maximum) and $2 > 0$ at $x = 0$ (minimum), so $x^* = 2$.

Subgame perfect equilibrium:

$$(x^*,\ \tilde{y}_1(x),\ \tilde{y}_2(y_1)) = \left(2,\ x,\ \frac{y_1}{2}\right) \tag{6.13}$$

On the equilibrium path: $x^* = 2$, $y_1^* = 2$, $y_2^* = 1$.

Here is some code to verify these calculations:

```
import sympy as sym

x, y1, y2 = sym.symbols("x y1 y2", real=True)
```

```python
u1 = 2 * y2 * y1 + x * y1 - y1**2 - sym.Rational(1, 3) * x**3
u2 = -(y1 - 2 * y2)**2

br2 = sym.solve(sym.diff(u2, y2), y2)[0]
br1 = sym.solve(sym.diff(u1, y1), y1)[0]
print("Player 2 BR:", br2)
print("Player 1 BR:", br1)

y1_eq = sym.solve(sym.Eq(y1, br1.subs(y2, br2)), y1)[0]
y2_eq = br2.subs(y1, y1_eq)
print("Simultaneous solution: y1 =", y1_eq, ", y2 =", y2_eq)

u1_reduced = u1.subs([(y1, y1_eq), (y2, y2_eq)])
x_star = sym.solve(sym.diff(u1_reduced, x), x)
print("Optimal x candidates:", x_star)
```

```
Player 2 BR: y1/2
Player 1 BR: x/2 + y2
Simultaneous solution: y1 = x , y2 = x/2
Optimal x candidates: [0, 2]
```

Solution 6.3: Solution to Exercise 3

Game 1. Nodes d and e are in the same information set, but d has action set $\{M, N\}$ while e has action set $\{A, B\}$. This violates the requirement that all nodes in an information set share the same action set. The game is not well-defined.

Game 2. Nodes b and c are in the same information set, but b has action set $\{C, D\}$ while c has action set $\{A, B\}$. The game is not well-defined for the same reason.

Game 3.

Subgames. Node c is a singleton and all its successors are terminals, so it initiates a subgame. Node b is a singleton and its only non-terminal successors, d and e, are both in the same information set $\{d, e\}$, which lies entirely within the subtree of b; so b also initiates a subgame.

Subgame at *c*. Player 2 chooses $C \to (10, 10)$ or $D \to (20, 20)$. Since $20 > 10$, Player 2 plays D.

Subgame at *b*. Player 2 chooses H (reaching node e) or T (reaching node d). Player 1, unable to distinguish d from e, plays a single action from $\{H, T\}$ at the information set. With rows as Player 2's actions at b and columns as Player 1's actions at $\{d, e\}$:

$$M_r = \begin{pmatrix} -2 & 1 \\ 2 & -1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & -1 \\ -2 & 1 \end{pmatrix} \tag{6.14}$$

There is no pure Nash equilibrium. For the mixed equilibrium, Player 1's indifference condition gives:

$$-2q + 2(1-q) = 2q - 2(1-q) \Longrightarrow q = \frac{1}{2} \tag{6.15}$$

and Player 2's indifference condition gives:

$$2p - (1-p) = -2p + (1-p) \Longrightarrow p = \frac{1}{3} \tag{6.16}$$

The unique Nash equilibrium of this subgame is Player 2 plays H with probability $\frac{1}{2}$ and Player 1 plays H with probability $\frac{1}{3}$. Player 1′s expected payoff at this equilibrium is 0.

Normal form of the whole game. The strategy sets are:

$$S_1 = \{AH,\ AT,\ BH,\ BT\}, \qquad S_2 = \{HC,\ HD,\ TC,\ TD\} \tag{6.17}$$

where the first letter of each strategy is the action at a (for Player 1) or b (for Player 2), and the second is the action at $\{d, e\}$ (for Player 1) or c (for Player 2). With rows ordered (AH, AT, BH, BT) and columns ordered (HC, HD, TC, TD):

$$M_r = \begin{pmatrix} 2 & 2 & -2 & -2 \\ -1 & -1 & 1 & 1 \\ 10 & 20 & 10 & 20 \\ 10 & 20 & 10 & 20 \end{pmatrix} \qquad M_c = \begin{pmatrix} -2 & -2 & 2 & 2 \\ 1 & 1 & -1 & -1 \\ 10 & 20 & 10 & 20 \\ 10 & 20 & 10 & 20 \end{pmatrix} \tag{6.18}$$

Pure Nash equilibria. For any Player 2 strategy, Player 1 prefers BH or BT (payoff 10 or 20) over any A-strategy. Given Player 1 plays B, Player 2 prefers D at c (payoff 20 over 10). There are four pure Nash equilibria: (BH, HD), (BH, TD), (BT, HD), (BT, TD).

Subgame perfection. Each pure Nash equilibrium fixes a pure strategy for Player 2 at b and Player 1 at $\{d, e\}$; none of these pure combinations are Nash equilibria in the subgame at b (which has no pure Nash equilibrium), so none of the four pure equilibria are subgame perfect.

The unique subgame perfect Nash equilibrium combines the mixed Nash equilibrium of the subgame at b with Player 1 choosing B at a (expected payoff $20 > 0$) and Player 2 choosing D at c:

$$\sigma_1 = \left(AH : 0,\ AT : 0,\ BH : \frac{1}{3},\ BT : \frac{2}{3}\right) \tag{6.19}$$

$$\sigma_2 = \left(HC : 0,\ HD : \frac{1}{2},\ TC : 0,\ TD : \frac{1}{2}\right) \tag{6.20}$$

Solution 6.4: Solution to Exercise 4

1. Payoff functions

Players choose locations $x_1, x_2 \in [0, 1]$. Customers are uniformly distributed and travel to the nearest seller. When $x_1 < x_2$:

- Player 1 serves customers in $\left[0, \frac{x_1+x_2}{2}\right]$, a fraction $\frac{x_1+x_2}{2}$ of the total.
- Player 2 serves customers in $\left[\frac{x_1+x_2}{2}, 1\right]$, a fraction $1 - \frac{x_1+x_2}{2}$.

When $x_1 = x_2$, each gets $\frac{1}{2}$. By symmetry, when $x_2 < x_1$, the roles reverse. Formally:

$$u_1(x_1, x_2) = \tag{6.21}$$

$$u_2(x_1, x_2) = 1 - u_1(x_1, x_2) \tag{6.22}$$

2. Player 2′s best response function

Player 2 observes x_1 and chooses x_2 to maximise their share. The payoff function has a discontinuity at $x_2 = x_1$, so we analyse three cases.

Case $x_1 < 1/2$. If $x_2 > x_1$, payoff is $1 - (x_1 + x_2)/2$, which is decreasing in x_2, so Player 2 prefers x_2 as small as possible, just above x_1, obtaining a payoff approaching $1 - x_1 > 1/2$. Setting $x_2 = x_1$ gives only $1/2 < 1 - x_1$. So Player 2 can do strictly better than $1/2$ and will not choose $x_2 = x_1$.

Case $x_1 = 1/2$. Any $x_2 > 1/2$ gives payoff $1 - (1/2 + x_2)/2 < 1/2$; any $x_2 < 1/2$ gives payoff $(1/2 + x_2)/2 < 1/2$. Only $x_2 = 1/2$ yields $1/2$. The unique best response is $x_2^*(1/2) = 1/2$.

Case $x_1 > 1/2$. By symmetry, Player 2 prefers x_2 just below x_1, obtaining payoff approaching $x_1 > 1/2$, again strictly better than $1/2$.

3. Subgame perfect equilibrium via backward induction

Player 1 anticipates Player 2's response. If $x_1 < 1/2$, Player 2 sets $x_2 = x_1 + \varepsilon$, giving Player 1 payoff $(x_1 + x_2)/2 \approx x_1 < 1/2$. If $x_1 > 1/2$, Player 2 sets $x_2 = x_1 - \varepsilon$, giving Player 1 payoff $1 - (x_2 + x_1)/2 \approx 1 - x_1 < 1/2$. Only $x_1 = 1/2$ guarantees Player 1 a payoff of $1/2$, with Player 2 best responding at $x_2 = 1/2$. The unique subgame perfect equilibrium is:

$$x_1^* = \frac{1}{2}, \quad x_2^* = \frac{1}{2} \tag{6.23}$$

Both players locate at the centre and share the market equally, each earning $\frac{1}{2}$.

7. Repeated Games

When the same game is played repeatedly, players can condition future behaviour on past actions, opening the door to cooperation that would be impossible in a one-shot interaction. This chapter shows how indefinite repetition can sustain cooperative outcomes, culminating in the Folk Theorem.

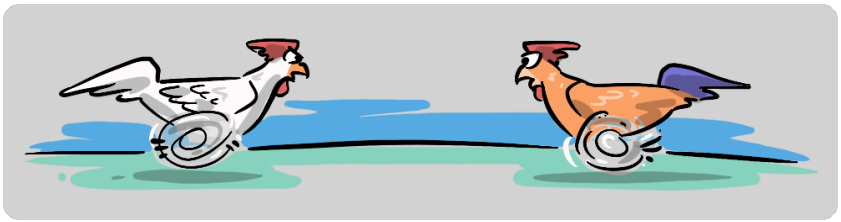


Figure 7.1: Two players square up in a contest of nerve. When such confrontations are repeated rather than played once, the prospect of meeting again changes what it is rational to do today.

7.1 Motivating Example: Construction Contractors

Consider two local construction firms, **Firm A** and **Firm B**, who regularly bid on municipal infrastructure projects.

Each quarter, the city may announce a new contract for firms to bid on: a school, a road, or a public facility. These firms can choose to:

- **Bid High (H)**: maintain high prices and sustain mutual profitability, or
- **Bid Low (L)**: undercut the other firm to win the project outright.

If both firms bid high, they enjoy good margins and may take turns winning contracts. If one undercuts while the other bids high, the undercutter wins the project and earns a higher profit, at the cost of undermining trust. If both bid low, a price war ensues, shrinking profits for both.

However, the number of projects is not fixed in advance. After each bidding round, there is a probability $\delta \in (0, 1)$ that **another project becomes available**. Thus, the game is repeated probabilistically, with continuation probability δ. The expected number of repetitions is $\frac{1}{1-\delta}$.

Each firm chooses an action in each round:

- H: Bid High
- L: Bid Low

The one-shot payoff matrix is given by:

$$M_r = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{7.1}$$

This is a classic example of a Prisoner's Dilemma: short-term incentives tempt each player to defect (bid low), but long-term cooperation (bid high) could yield better payoffs.

The remainder of this chapter will explore:

- How cooperation can be sustained using history-dependent strategies.
- What conditions on δ allow for equilibrium cooperation.
- The formal statement and implications of the **Folk Theorem**.

7.2 Theory

7.2.1 Definition: repeated game

Given a two player game $(A, B) \in \mathbb{R}^{m \times n^2}$, referred to as a stage game, a T-stage repeated game is a game in which players play that stage game for $T > 0$ repetitions. Players make decisions based on the full history of play over all the repetitions.

7.2.1.1 Example: Counting leaves in repeated games

Consider the following stage games and values of T. How many leaves would the extensive form representation of the repeated game have?

1.

$$M_r = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix} \qquad T = 2 \tag{7.2}$$

The initial play of the game has 4 outcomes (2 actions each). Each of these leads to 4 more in the second stage. Total: $4 \times 4 = 16$ leaves.

2.

$$M_r = \begin{pmatrix} 0 & 1 \\ -1 & 3 \end{pmatrix} \qquad M_c = -M_r \qquad T = 2 \tag{7.3}$$

Same as (1): $4 \times 4 = 16$ leaves.

7.2.2 Definition: Strategies in a repeated game

A strategy for a player in a repeated game is a mapping from all possible histories of play to a probability distribution over the action set of the stage game.

7.2.2.1 Example: Validity of repeated game strategies

For the Coordination Game with $T = 2$ determine whether the following strategy pairs are valid, and if so, what outcome they lead to.

1. Row player:

$$\begin{aligned} (\varnothing, \varnothing) &\to C \\ (S, S) &\to C \\ (S, C) &\to C \\ (C, S) &\to S \\ (C, C) &\to S \end{aligned} \tag{7.4}$$

Column player:

$$\begin{aligned} (\varnothing, \varnothing) &\to S \\ (S, S) &\to C \\ (S, C) &\to C \\ (C, S) &\to S \\ (C, C) &\to S \end{aligned} \tag{7.5}$$

Valid strategy pair. Outcome: $(3, 2)$ (corresponds to O_9 in the extensive form).

2. Row player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to C \\
(S, S) &\to C \\
(C, S) &\to S \\
(C, C) &\to S
\end{aligned}
\tag{7.6}
$$

Column player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to S \\
(S, S) &\to C \\
(S, C) &\to C \\
(C, S) &\to S \\
(C, C) &\to S
\end{aligned}
\tag{7.7}
$$

Invalid: the row player does not define an action for history (S, C).

3. Row player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to C \\
(S, S) &\to C \\
(C, S) &\to S \\
(S, C) &\to S \\
(C, C) &\to S
\end{aligned}
\tag{7.8}
$$

Column player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to S \\
(S, S) &\to C \\
(S, C) &\to C \\
(C, S) &\to \alpha \\
(C, C) &\to S
\end{aligned}
\tag{7.9}
$$

Invalid: column player uses an invalid action α.

4. Row player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to S \\
(S, S) &\to C \\
(C, S) &\to S \\
(S, C) &\to C \\
(C, C) &\to S
\end{aligned}
\tag{7.10}
$$

Column player:

$$
\begin{aligned}
(\varnothing, \varnothing) &\to S \\
(S, S) &\to C \\
(S, C) &\to C \\
(C, S) &\to S \\
(C, C) &\to S
\end{aligned}
\tag{7.11}
$$

Valid strategy pair. Outcome: $(5, 5)$ (corresponds to O_4 in the extensive form).

7.2.3 Theorem: Subgame perfection of sequence of stage Nash profiles

For any repeated game, any sequence of stage Nash profiles gives the outcome of a subgame perfect Nash equilibrium.

Where by stage Nash profile we refer to a strategy profile that is a Nash Equilibrium in the stage game.

7.2.3.1 Proof

Fix a stage Nash profile for each period: in period k every player i plays the action $\tilde{s}_i^{(k)}$ of some Nash equilibrium of the stage game, regardless of the history of play. This is a well-defined strategy profile for the repeated game, and it is history-independent, so it prescribes a stage Nash profile in every subgame.

We show it is subgame perfect by backward induction on the period. Total payoffs are the (discounted) sum of the stage payoffs, so in the final period T each player faces exactly the stage game; since $\tilde{s}^{(T)}$ is a stage Nash profile, no player can gain by deviating in period T. Suppose the prescribed continuation from period $k+1$ onwards is a Nash equilibrium of every subgame it initiates. In period k a player's total payoff is their period-k stage payoff plus a continuation payoff that, because the strategies ignore history, does not depend on the period-k action. Optimising therefore reduces to the stage game, in which $\tilde{s}^{(k)}$ is a Nash profile, so again no player can gain by deviating. By the one-shot deviation principle no player has any profitable deviation in any subgame, and the profile is subgame perfect.

7.2.3.2 Example

Consider the following stage game:

$$M_r = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 2 & 4 \\ 1 & 2 & 0 \end{pmatrix} \tag{7.12}$$

There are two Nash equilibria in action space for this stage game:

$$(r_1, c_3) \qquad (r_2, c_2) \tag{7.13}$$

For $T = 2$ we have 4 possible outcomes that correspond to the outcome of a subgame perfect Nash equilibria:

$$(r_1 r_1, c_3 c_3) \text{ giving utility vector : } (2, 8) \tag{7.14}$$

$$(r_1 r_2, c_3 c_2) \text{ giving utility vector : } (2, 6) \tag{7.15}$$

$$(r_2 r_1, c_2 c_3) \text{ giving utility vector : } (2, 6) \tag{7.16}$$

$$(r_2 r_2, c_2 c_2) \text{ giving utility vector : } (2, 4) \tag{7.17}$$

Importantly, not all subgame Nash equilibria outcomes are of the above form.

7.2.4 Reputation

In a repeated game it is possible for players to encode reputation and trust in their strategies.

7.2.4.1 Example: Reputation in a repeated game

Consider the following stage game with $T = 2$:

$$A = \begin{pmatrix} 0 & 6 & 1 \\ 1 & 7 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 3 & 1 \\ 1 & 0 & 1 \end{pmatrix} \tag{7.18}$$

Through inspection it is possible to verify that the following strategy pair is a Nash equilibrium.

For the row player:

$$\begin{aligned}
(\varnothing, \varnothing) &\to r_1 \\
(r_1, c_1) &\to r_2 \\
(r_1, c_2) &\to r_2 \\
(r_1, c_3) &\to r_2 \\
(r_2, c_1) &\to r_2 \\
(r_2, c_2) &\to r_2 \\
(r_2, c_3) &\to r_2
\end{aligned} \tag{7.19}$$

For the column player:

$$\begin{aligned}
(\varnothing, \varnothing) &\to c_2 \\
(r_1, c_1) &\to c_3 \\
(r_2, c_1) &\to c_1 \\
(r_1, c_2) &\to c_3 \\
(r_2, c_2) &\to c_1 \\
(r_1, c_3) &\to c_3 \\
(r_2, c_3) &\to c_1
\end{aligned} \tag{7.20}$$

This pair of strategies corresponds to the following scenario:

The row player plays r_1 and the column player plays c_2 in the first stage. The row player plays r_2 and the column player plays c_3 in the second stage.

Note that if the row player deviates and plays r_2 in the first stage, then the column player will play c_1 in the second stage.

If both players play these strategies, their utilities are: $(11, 4)$, which is better **for both players** than the utilities at any sequence of stage Nash equilibria in action space.

But is this a Nash equilibrium? To find out, we investigate whether either player has an incentive to deviate.

1. If the row player deviates, they would only be rational to do so in the first stage. If they did, they would gain 1 in that stage but lose 4 in the second stage. Thus they have no incentive to deviate.
2. If the column player deviates, they would only do so in the first stage and gain no utility.

Thus, this strategy pair **is a Nash equilibrium** and evidences how reputation can be built and cooperation can emerge from complex dynamics.

7.2.5 Definition: Infinitely Repeated Game with Discounting

Figure 7.2: A bird in the hand is worth two in the bush. The discount factor δ captures exactly this preference for payoffs received now over those promised later.

Given a two player game $(A, B) \in \mathbb{R}^{m \times n^2}$, referred to as a stage game, an infinitely repeated game with discounting factor δ is a game in which players play that stage game an infinite amount of times and gain the following utility:

$$U(s_r, s_c) = \sum_{i=0}^{\infty} \delta^i u(s_r(i), s_c(i)) \tag{7.21}$$

where $u(s_r(i), s_c(i))$ denotes the utility obtained in the stage game for actions $s_r(i)$ and $s_c(i)$ which are the actions given by strategies s_r and s_c at stage i.

Note

The interpretation of δ can be twofold. Firstly it can be thought of as a common economic tool to reduce value of future gains. Secondly it can be thought of as a probability of the game ending.

7.2.5.1 Example:

Consider Motivating Example: Construction Contractors and assume Firm A plans to undercut at all contracts: bidding low. Firm B plans to start by cooperating (bidding high) but if Firm A ever undercuts then Firm B will undercut for all subsequent contracts.

$$\begin{aligned} U_A(s_r, s_c) &= \sum_{i=0}^{\infty} \delta^i u_A(s_r(i), s_c(i)) \\ &= u_A(L, H) + \sum_{i=1}^{\infty} \delta^i u_A(L, L) \\ &= u_A(L, H) + u_A(L, L) \sum_{i=1}^{\infty} \delta^i \\ &= u_A(L, H) + u_A(L, L) \frac{\delta}{1-\delta} \end{aligned} \tag{7.22}$$

and

$$\begin{aligned} U_B(s_r, s_c) &= \sum_{i=0}^{\infty} \delta^i u_B(s_r(i), s_c(i)) \\ &= u_B(L, H) + \sum_{i=1}^{\infty} \delta^i u_B(L, L) \\ &= u_B(L, H) + u_B(L, L) \sum_{i=1}^{\infty} \delta^i \\ &= u_B(L, H) + u_B(L, L) \frac{\delta}{1-\delta} \end{aligned} \tag{7.23}$$

Replacing the values from the stage game this gives:

$$U_A(s_r, s_c) = 5 + \frac{\delta}{1-\delta} \qquad U_B(s_r, s_c) = 0 + \frac{\delta}{1-\delta} \tag{7.24}$$

7.2.6 Definition: Average utility

If we interpret δ as the probability of the repeated game not ending then the *average* length of the game is:

$$\bar{T} = \frac{1}{1-\delta} \tag{7.25}$$

We can use this to define the **average payoff** per stage:

$$\bar{U}_i(r,c) = (1-\delta)U_i(r,c) \tag{7.26}$$

7.2.6.1 Example:

Consider 3 strategies for Motivating Example: Construction Contractors:

- S_c Always cooperate: bid high on all contracts.
- S_d Always defect: bid low on all contracts.
- S_g Grudger: bid high until facing a low bid and then bid low for ever.

For $\delta \in (1/4, 3/4)$:

1. Obtain the 3 by 3 Normal form game using average utility.
2. Check if mutual cooperation is a Nash equilibrium for both games.

We construct the Normal Form game representation with the 3 strategies becoming the action space (as described in Mapping Extensive Form Games to Normal Form).

We see that if S_c is played against S_g then both players cooperate at each stage giving:

$$\bar{U}_r(S_c,S_c) = \bar{U}_r(S_g,S_g) = \bar{U}_r(S_g,S_c) = \bar{U}_r(S_c,S_g) = (1-\delta)\frac{3}{1-\delta} = 3 \tag{7.27}$$

For S_d and S_c we have:

$$\bar{U}_r(S_d,S_c) = 5 \tag{7.28}$$

$$\bar{U}_r(S_d,S_d) = 1 \tag{7.29}$$

$$\bar{U}_r(S_c,S_d) = 0 \tag{7.30}$$

For S_g and S_d we repeat the calculations of Example: to obtain:

$$\bar{U}_r(S_g,S_d) = (1-\delta)\left(0 + \sum_{i=1}^{\infty}\delta^i\right) = (1-\delta)\frac{\delta}{1-\delta} = \delta \tag{7.31}$$

and

$$\bar{U}_r(S_d,S_g) = (1-\delta)\left(5 + \sum_{i=1}^{\infty}\delta^i\right) = (1-\delta)5 + (1-\delta)\frac{\delta}{1-\delta} = 5 - 4\delta \tag{7.32}$$

Using the action space ordered as: $\{S_c, S_d, S_g\}$ this gives:

$$M_r = \begin{pmatrix} 3 & 0 & 3 \\ 5 & 1 & 5-4\delta \\ 3 & \delta & 3 \end{pmatrix} \tag{7.33}$$

The game is symmetric so:

$$M_c = M_r^T \tag{7.34}$$

For $\delta = 1/4$ this gives:

$$M_r = \begin{pmatrix} 3 & 0 & 3 \\ \underline{5} & \underline{1} & \underline{4} \\ 3 & \frac{1}{4} & 3 \end{pmatrix} \tag{7.35}$$

The game is symmetric so:

$$M_c = \begin{pmatrix} 3 & \underline{5} & 3 \\ 0 & \underline{1} & \frac{1}{4} \\ 3 & \underline{4} & 3 \end{pmatrix} \tag{7.36}$$

We see that s_c and s_g are both dominated by s_d thus mutual cooperation is not a Nash equilibrium.

For $\delta = 3/4$ this gives:

$$M_r = \begin{pmatrix} 3 & 0 & \underline{3} \\ \underline{5} & \underline{1} & 2 \\ 3 & \frac{3}{4} & \underline{3} \end{pmatrix} \tag{7.37}$$

The game is symmetric so:

$$M_c = \begin{pmatrix} 3 & \underline{5} & 3 \\ 0 & \underline{1} & \frac{3}{4} \\ \underline{3} & 2 & \underline{3} \end{pmatrix} \tag{7.38}$$

We see that mutual cooperation now is a Nash equilibrium because s_g is now a best response to s_g.

The fact that cooperation can be stable for a higher value of δ is in fact a theorem that holds in the general case.

We need one final definition to describe what we imply:

7.2.7 Definition of individually rational payoffs

Individually rational payoffs are average payoffs that exceed the stage game Nash equilibrium payoffs for both players.

As an example consider the plot corresponding to a repeated Prisoner's Dilemma shown in Figure 7.3.

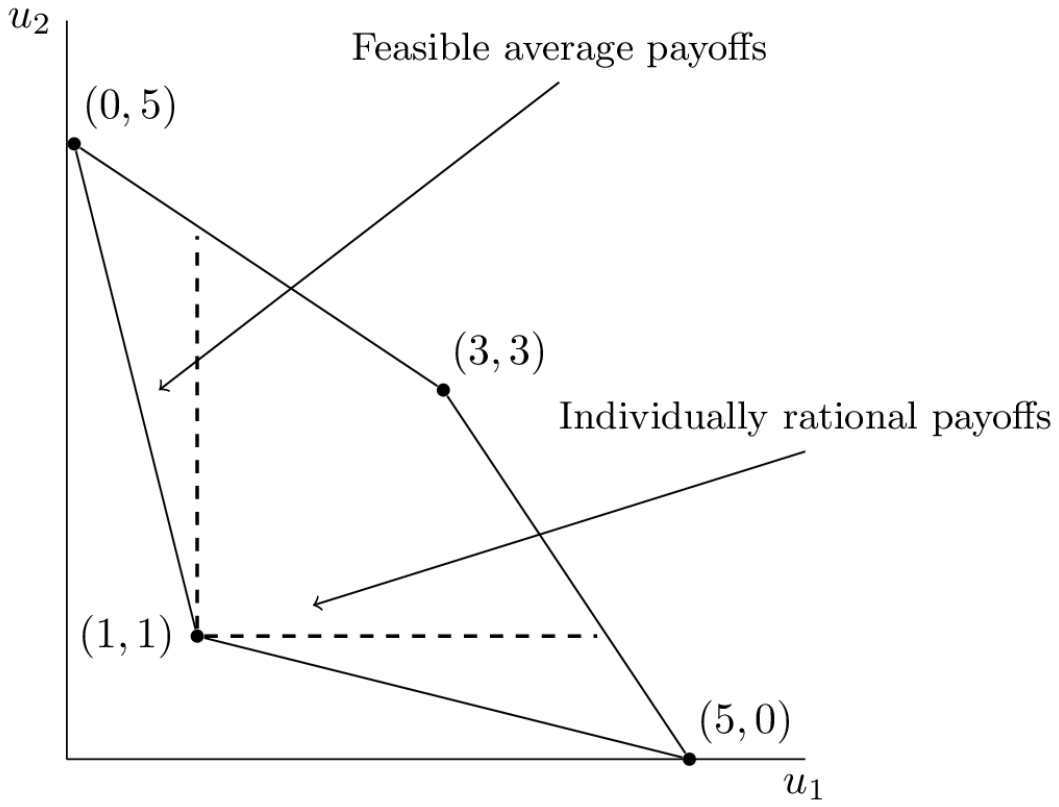


Figure 7.3: The convex hull of payoffs for the Prisoner's Dilemma.

The feasible average payoffs correspond to the feasible payoffs in the stage game. The individually rational payoffs show the payoffs that are **better for both players** than the stage Nash equilibrium.

The following theorem states that we can choose a particular discount rate that for which there exists a subgame perfect Nash equilibrium that would give any individually rational payoff pair!

7.2.8 Theorem: Folk Theorem

Let (u_1^*, u_2^*) be a pair of Nash equilibrium payoffs for a stage game. For every individually rational pair (v_1, v_2) there exists $\bar{\delta}$ such that for all $1 > \delta > \bar{\delta} > 0$ there is a subgame perfect Nash equilibrium with payoffs (v_1, v_2).

7.2.8.1 Proof

Let (σ_1^*, σ_2^*) be the stage Nash profile that yields (u_1^*, u_2^*). Now assume that playing $\bar{\sigma}_1 \in \Delta\mathcal{A}_1$ and $\bar{\sigma}_2 \in \Delta\mathcal{A}_2$ in every stage gives (v_1, v_2) (an individual rational payoff pair).

Consider the following strategy:

> "Begin by using $\bar{\sigma}_i$ and continue to use $\bar{\sigma}_i$ as long as both players use the agreed strategies. If any player deviates: use σ_i^* for all future stages."

We begin by proving that the above is a Nash equilibrium.

Without loss of generality if player 1 deviates to ${\sigma_1}' \in \Delta S_1$ such that $u_1({\sigma_1}', \bar{\sigma}_2) > v_1$ in stage k then:

$$U_1^{(k)} = \sum_{t=1}^{k-1} \delta^{t-1} v_1 + \delta^{k-1} u_1(\sigma_1', \bar{\sigma}_2) + u_1^* \left(\frac{1}{1-\delta} - \sum_{t=1}^{k} \delta^{t-1} \right) \tag{7.39}$$

Recalling that player 1 would receive v_1 in every stage with no deviation, the biggest gain to be made from deviating is if player 1 deviates in the first stage (all future gains are more heavily discounted). Thus if we can find $\bar{\delta}$ such that $\delta > \bar{\delta}$ implies that $U_1^{(1)} \leq \frac{v_1}{1-\delta}$ then player 1 has no incentive to deviate.

$$\begin{aligned} U_1^{(1)} = u_1(\sigma_1', \bar{\sigma}_2) + u_1^* \frac{\delta}{1-\delta} &\leq \frac{v_1}{1-\delta} \\ (1-\delta) u_1(\sigma_1', \bar{\sigma}_2) + u_1^* \delta &\leq v_1 \\ u_1(\sigma_1', \bar{\sigma}_2) - v_1 &\leq \delta (u_1(\sigma_1', \bar{\sigma}_2) - u_1^*) \end{aligned} \tag{7.40}$$

The most profitable one-stage deviation is the stage best response, so write $\hat{u}_1 = \max_{\sigma_1'} u_1(\sigma_1', \bar{\sigma}_2)$ for its payoff. As (v_1, v_2) is individually rational we have $v_1 > u_1^*$, and the map $x \mapsto \frac{x - v_1}{x - u_1^*}$ is increasing for $x > v_1 > u_1^*$, so the binding threshold is the one for the best deviation. Taking $\bar{\delta}_1 = \frac{\hat{u}_1 - v_1}{\hat{u}_1 - u_1^*} \in (0, 1)$ ensures player 1 has no profitable deviation whenever $\delta > \bar{\delta}_1$. Repeating the argument for player 2 gives $\bar{\delta}_2$, and setting $\bar{\delta} = \max\left(\bar{\delta}_1, \bar{\delta}_2\right)$ shows that the prescribed strategy is a Nash equilibrium for all $\delta > \bar{\delta}$.

By construction this strategy is also a subgame perfect Nash equilibrium. Given any history **both** players will act in the same way and no player will have an incentive to deviate:

- If we consider a subgame just after any player has deviated from $\bar{\sigma}_i$ then both players use σ_i^*.
- If we consider a subgame just after no player has deviated from $\bar{\sigma}_i$ then both players continue to use $\bar{\sigma}_i$.

Note

The punishment used here is reversion to a stage Nash equilibrium, so the payoffs that can be sustained are those that strictly beat a Nash payoff of the stage game; this is the sense in which we use individually rational in this chapter. A stronger version of the Folk Theorem, due to Fudenberg and Maskin (Fudenberg & Maskin, 1986), punishes a deviator with their **minmax** value rather than a Nash payoff. Since the minmax value is never larger than a Nash payoff, this sustains every payoff above the minmax, a larger region. That version requires more elaborate punishment strategies and is a natural extension of the treatment here.

7.3 Exercises

Exercise 7.1:

Recalling that a subgame perfect equilibrium for a finitely repeated game must involve a stage Nash equilibrium in the final stage, attempt to identify a subgame perfect equilibrium for the repeated game that is not a simple repetition of stage Nash profiles for each of the following games.

Game 1

Let

$$M_r = \begin{pmatrix} 4 & 7 \\ 1 & 4 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 3 & 6 \\ 1 & 3 \end{pmatrix} \tag{7.41}$$

Game 2

Let

$$M_r = \begin{pmatrix} 5 & 0 \\ 0 & 1 \\ 3 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 4 & 3 \\ 3 & 4 \\ 6 & 3 \end{pmatrix} \tag{7.42}$$

Game 3

Let

$$M_r = \begin{pmatrix} 1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 2 & 3 & 1 \\ 0 & -1 & 1 \end{pmatrix} \tag{7.43}$$

Exercise 7.2:

For a general stage game with $(M_r, M_c) \in \mathbb{R}^{(m \times n)^2}$ identify the size of the action space for the repeated game for each player when:

- $m = 2, n = 2$ and $T = 2$.
- General m, n and $T = 3$.
- General m, n, T.

Exercise 7.3:

Consider the following stage game:

$$M_r = \begin{pmatrix} -1 & 3 \\ -2 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & -7 \\ 6 & 2 \end{pmatrix} \tag{7.44}$$

1. For $\delta = \frac{1}{3}$, compute the average payoffs for the strategies S_D: "always play the first row" and S_C: "always play the second row".
2. Plot the feasible average payoff space and the individually rational payoff space.
3. Using the Folk Theorem, determine whether there exists a value of δ that supports a subgame perfect equilibrium with average payoffs:
   - $(3/2, 3/2)$
   - $(0, 3)$
   - $(2, 6)$
   - $(2, 0)$

Exercise 7.4:

Consider the standard Prisoner's Dilemma:

$$M_r = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{7.45}$$

Suppose players repeatedly play this game using one of the strategies:

1. **Tit For Tat**: starts by cooperating, then repeats the opponent's previous action.

2. **Alternator**: starts by cooperating, then alternates between cooperation and defection.

Obtain the normal form representation of the repeated game for the case of an infinitely repeated game with discount factor δ.

Determine the Nash equilibria in action space and interpret this result.

7.4 Programming

7.4.1 Using Nashpy to generate repeated games

The Nashpy library has support for generating repeated games. Let us generate the repeated game obtained from the Prisoners Dilemma with $T = 2$:

```
import nashpy.repeated_games

import nashpy as nash
import numpy as np

M_r = np.array([[3, 0], [5, 1]])
M_c = M_r.T
prisoners_dilemma = nash.Game(M_r, M_c)
repeated_pd = nash.repeated_games.obtain_repeated_game(game=prisoners_dilemma,
repetitions=2)
print(repeated_pd)
```

```
Bi matrix game with payoff matrices:

Row player:
[[ 6.  6.  6. ...  3.  0.  0.]
 [ 6.  6.  6. ...  3.  0.  0.]
 [ 6.  6.  6. ...  3.  0.  0.]
 ...
 [ 8.  8.  8. ...  2.  6.  2.]
 [10. 10. 10. ...  1.  4.  1.]
 [10. 10. 10. ...  2.  6.  2.]]

Column player:
[[ 6.  6.  6. ...  8. 10. 10.]
 [ 6.  6.  6. ...  8. 10. 10.]
 [ 6.  6.  6. ...  8. 10. 10.]
 ...
 [ 3.  3.  3. ...  2.  1.  2.]
 [ 0.  0.  0. ...  6.  4.  6.]
 [ 0.  0.  0. ...  2.  1.  2.]]
```

Warning

The action space for repeated games can become incredibly large even for low values of repetitions. This might result in games that cannot be generated computationally.

We can directly obtain the action space of the row player for a given repeated game as a python generator:

```
strategies = nash.repeated_games.obtain_strategy_space(A=M_r, repetitions=2)
print(f"Number of row player strategies: {len(list(strategies))}")
```

```
Number of row player strategies: 32
```

To obtain the action space for the column player use `A=M_c.T`:

```
strategies = nash.repeated_games.obtain_strategy_space(A=M_c.T, repetitions=2)
print(f"Number of column player strategies: {len(list(strategies))}")
```

```
Number of column player strategies: 32
```

7.5 Notable research

The Folk Theorem presented in this chapter has a rich theoretical lineage. The foundational result is due to (Friedman, 1971) (with a correction in (Friedman, 1973)), with the general formulation appearing in (Fudenberg & Maskin, 1986). These results established the theoretical basis for understanding how cooperation can be sustained in long-run interactions.

(Young, 1993) showed how social conventions and norms can emerge from adaptive play in repeated settings, providing an evolutionary foundation for Folk Theorem-type outcomes.

7.6 Conclusion

Repeated games introduce a rich strategic landscape where history matters and reputation can sustain cooperation. Unlike one-shot interactions, repeated games allow for long-term incentives to shape behaviour, often leading to outcomes that dominate those of stage game equilibria.

This chapter covered the formal definitions of repeated and infinitely repeated games, strategy spaces based on full histories, subgame perfection through sequences of Nash profiles, and the pivotal role of discounting. Through examples and the Folk Theorem, we saw how cooperation can emerge, even in environments like the Prisoner's Dilemma, when players value the future sufficiently.

Table 7.1 summarises the key concepts.

Concept	Description
Repeated game	A game consisting of multiple repetitions of a fixed stage game
Strategy in repeated game	A mapping from history of play to actions
Subgame perfect equilibrium	An equilibrium where players play a Nash profile in every subgame
Reputation	Players may cooperate to sustain credibility and trust over time
Discount factor (δ)	Models time preference or probability of continuation
Average utility	$(1-\delta)$ times the discounted sum of stage payoffs
Individually rational payoffs	Average payoffs that exceed stage game Nash equilibrium payoffs
Folk Theorem	Any individually rational payoff can be sustained given δ is high enough

Table 7.1: Summary of key concepts in repeated games

Attention

Repeated games illustrate how cooperation can be rational, even when short-term incentives push toward defection. Through strategies that condition on history and the presence of credible threats, players can build trust and sustain mutually beneficial outcomes. The Folk Theorem formalises this by showing that *any* payoff better than the stage game Nash equilibrium can be supported, provided the players are sufficiently patient.

7.7 Solutions

Solution 7.1: Solution to Exercise 1

Recall that in a finitely repeated game the final stage must be played as a stage Nash equilibrium. To sustain a non-Nash outcome in the first stage we need **multiple** stage Nash equilibria: one can be used as a reward and another as a punishment.

Game 1

$$M_r = \begin{pmatrix} 4 & 7 \\ 1 & 4 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 3 & 6 \\ 1 & 3 \end{pmatrix} \tag{7.46}$$

First find the stage Nash equilibria. Checking best responses:

- (r_1, c_1): Row gets 4 vs 1 (prefers r_1); Column gets 3 vs 6 (prefers c_2). Not NE.
- (r_1, c_2): Row gets 7 vs 4 (prefers r_1); Column gets 6 vs 3 (prefers c_2). NE.
- (r_2, c_1): Row gets 1 vs 4 (prefers r_1). Not NE.
- (r_2, c_2): Row gets 4 vs 7 (prefers r_1). Not NE.

There is only **one** pure strategy Nash equilibrium: (r_1, c_2). With a unique stage Nash equilibrium, there is no second equilibrium to use as a punishment. Thus it is **not possible** to sustain a non-stage-Nash outcome as a subgame perfect equilibrium for any finite repetition.

Game 2

$$M_r = \begin{pmatrix} 5 & 0 \\ 0 & 1 \\ 3 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 4 & 3 \\ 3 & 4 \\ 6 & 3 \end{pmatrix} \tag{7.47}$$

Find stage Nash equilibria by checking best responses in action space:

- (r_1, c_1): Row: $5 \geq 0 \geq 3$? Row prefers r_1. Column: 4 vs 3, prefers c_1. Is (r_1, c_1) a NE? Row BR to c_1: r_1 (5>0>3). Column BR to r_1: c_1 (4>3). **NE**.
- (r_2, c_2): Row BR to c_2: r_2 (1>0>0). Column BR to r_2: c_2 (4>3). **NE**.
- (r_3, c_1): Row BR to c_1: r_1 (5>0>3). Not NE.

So there are (at least) two pure Nash equilibria: (r_1, c_1) with payoffs $(5, 4)$ and (r_2, c_2) with payoffs $(1, 4)$.

With two Nash equilibria we can construct a subgame perfect equilibrium using (r_1, c_1) as a reward and (r_2, c_2) as a punishment to sustain (r_3, c_1) in stage 1.

Incentive constraints. Row's payoff under cooperation: $3 + 5 = 8$. Row's best deviation in stage 1 given column plays c_1 is r_1, yielding $5 + 1 = 6$. Since $8 > 6$, row does not deviate. Column's payoff under cooperation: $6 + 4 = 10$. Column's best deviation given row plays r_3 is c_2, yielding $3 + 4 = 7$. Since $10 > 7$, column does not deviate.

The subgame perfect equilibrium is given by the following two strategies:

Row's strategy s_r:

- Stage 1: play r_3.
- Stage 2: play r_1 if the stage 1 outcome was (r_3, c_1); play r_2 otherwise.

Column's strategy s_c:

- Stage 1: play c_1.
- Stage 2: play c_1 if the stage 1 outcome was (r_3, c_1); play c_2 otherwise.

This yields total payoffs $(3+5,\ 6+4) = (8, 10)$, which is not a repetition of stage Nash profiles.

Verifying neither player has an incentive to deviate.

In stage 2 both players are prescribed a stage Nash equilibrium, so neither can gain by deviating there regardless of the stage 1 history.

In stage 1, given the opponent follows s_c (resp. s_r):

- Row receives $3+5=8$ by following s_r. The best deviation is r_1, which gives 5 in stage 1 but triggers the punishment (r_2, c_2) in stage 2, yielding $5+1=6<8$. Row does not deviate.
- Column receives $6+4=10$ by following s_c. The best deviation is c_2, which gives 3 in stage 1 but triggers (r_2, c_2) in stage 2, yielding $3+4=7<10$. Column does not deviate.

Therefore (s_r, s_c) is a Nash equilibrium.

Game 3

$$M_r = \begin{pmatrix} 1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 2 & 3 & 1 \\ 0 & -1 & 1 \end{pmatrix} \tag{7.48}$$

Find stage Nash equilibria:

- (r_1, c_2): Row BR to c_2: r_1 ($0 > -1$). Column BR to r_1: c_2 ($3 > 2 > 1$). **NE**, payoffs $(0, 3)$.
- (r_2, c_3): Row BR to c_3: r_2 ($0 > -1$). Column BR to r_2: c_1 or c_3 ($1 > -1$, tie). **NE**, payoffs $(0, 1)$.

With two stage Nash equilibria we use (r_1, c_2) as a reward (column's preferred equilibrium) and (r_2, c_3) as a punishment to sustain (r_1, c_1) in stage 1.

Incentive constraints. Row already plays their best response to c_1 in stage 1, so row has no incentive to deviate. Column's payoff under cooperation: $2+3=5$. Column's best deviation given row plays r_1 is c_2, yielding $3+1=4$. Since $5>4$, column does not deviate.

The subgame perfect equilibrium is given by the following two strategies:

Row's strategy s_r:

- Stage 1: play r_1.
- Stage 2: play r_1 if the stage 1 outcome was (r_1, c_1); play r_2 otherwise.

Column's strategy s_c:

- Stage 1: play c_1.
- Stage 2: play c_2 if the stage 1 outcome was (r_1, c_1); play c_3 otherwise.

This yields total payoffs $(1+0,\ 2+3) = (1, 5)$, which is not a repetition of stage Nash profiles.

Verifying neither player has an incentive to deviate.

In stage 2 both players are prescribed a stage Nash equilibrium, so neither can gain by deviating there regardless of the stage 1 history.

In stage 1, given the opponent follows s_c (resp. s_r):

- Row is already playing their best response to c_1 in stage 1 (r_1 gives $1 > -1$), so row has no incentive to deviate regardless of stage 2 consequences.
- Column receives $2 + 3 = 5$ by following s_c. The best deviation is c_2, which gives 3 in stage 1 but triggers the punishment (r_2, c_3) in stage 2, yielding $3 + 1 = 4 < 5$. Column does not deviate.

Therefore (s_r, s_c) is a Nash equilibrium.

Solution 7.2: Solution to Exercise 2

A strategy in a repeated game maps every possible history of play to an action. We count the number of pure strategies for the row player in each case, then identify the general pattern.

Case 1: $m = 2$, $n = 2$, $T = 2$.

At stage 1 there is one history (the empty history), and row must choose one of $m = 2$ actions: 2 choices.

At stage 2 the history is the single stage-1 outcome. There are $mn = 4$ possible stage-1 outcomes, so 4 possible histories, and row must specify an action for each: $2^4 = 16$ choices.

The total number of pure strategies is:

$$2 \times 2^4 = 2^1 \cdot 2^4 = 2^5 = 32 \tag{7.49}$$

Case 2: General m, n, $T = 3$.

At stage 1: 1 history, m choices.

At stage 2: mn possible stage-1 outcomes, so mn histories. Row must specify an action for each: m^{mn} choices.

At stage 3: $(mn)^2$ possible two-stage histories. Row must specify an action for each: $m^{(mn)^2}$ choices.

The total number of pure strategies is:

$$m^1 \cdot m^{mn} \cdot m^{(mn)^2} = m^{1+mn+(mn)^2} \tag{7.50}$$

Case 3: General m, n, T.

At stage t there are $(mn)^{t-1}$ possible histories, requiring $m^{(mn)^{t-1}}$ action choices. The total number of pure strategies is:

$$\prod_{t=1}^{T} m^{(mn)^{t-1}} = m^{\sum_{t=0}^{T-1}(mn)^t} = m^{\frac{(mn)^T - 1}{mn-1}} \tag{7.51}$$

By the same reasoning, the column player has $n^{\frac{(mn)^T - 1}{mn-1}}$ pure strategies.

Here are some scenarios:

```python
import sympy as sym

m, n, T_sym = sym.symbols("m n T", positive=True, integer=True)

# Specific case m=2, n=2, T=2
result_22 = int(2 ** ((4**2 - 1) // (4 - 1)))
print(f"m=2, n=2, T=2: row player has {result_22} pure strategies")

# m=2, n=2, T=3
exponent_T3 = 1 + 4 + 16
result_223 = 2**exponent_T3
print(f"m=2, n=2, T=3: row player has {result_223} pure strategies")
```

```
m=2, n=2, T=2: row player has 32 pure strategies
m=2, n=2, T=3: row player has 2097152 pure strategies
```

Solution 7.3: Solution to Exercise 3

Consider the stage game:

$$M_r = \begin{pmatrix} -1 & 3 \\ -2 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & -7 \\ 6 & 2 \end{pmatrix} \tag{7.52}$$

1. Average payoffs for S_D (always row 1) and S_C (always row 2), $\delta = 1/3$.

If both players use S_D (always r_1/c_1), the stage payoff is $(-1, 1)$ every round:

$$\bar{U}_r(S_D, S_D) = (1-\delta)(-1)\sum_{i=0}^{\infty}\delta^i = (1-\delta)\frac{-1}{1-\delta} = -1 \tag{7.53}$$

$$\bar{U}_c(S_D, S_D) = 1 \tag{7.54}$$

If both use S_C (always r_2/c_2), stage payoff is $(2, 2)$:

$$\bar{U}_r(S_C, S_C) = (1-\delta)\frac{2}{1-\delta} = 2, \qquad \bar{U}_c(S_C, S_C) = 2 \tag{7.55}$$

If row plays S_D and column plays S_C (row always plays r_1, column always plays c_2): Stage payoff is $(3, -7)$:

$$\bar{U}_r(S_D, S_C) = 3, \qquad \bar{U}_c(S_D, S_C) = -7 \tag{7.56}$$

If row plays S_C and column plays S_D: Row always plays r_2, column always plays c_1: stage payoff is $(-2, 6)$:

$$\bar{U}_r(S_C, S_D) = -2, \qquad \bar{U}_c(S_C, S_D) = 6 \tag{7.57}$$

The Nash equilibria of the stage game must be identified to proceed. We check:

- (r_1, c_1): Row BR to c_1: $-1 > -2$ so r_1. Column BR to r_1: $1 > -7$ so c_1. **NE** with payoff $(-1, 1)$.
- (r_2, c_2): Row BR to c_2: $2 > 3$? No, r_1 gives 3. Not NE.

So the unique pure Nash equilibrium of the stage game is (r_1, c_1) with payoffs $(-1, 1)$.

```
import nashpy as nash
import numpy as np

M_r = np.array([[-1, 3], [-2, 2]])
M_c = np.array([[1, -7], [6, 2]])
game = nash.Game(M_r, M_c)
print("Stage Nash equilibria:")
for eq in game.support_enumeration():
    print(eq)
print()

delta = 1 / 3
print(f"delta = {delta}")
print(f"Average payoff (S_D, S_D): row={-1}, col={1}")
print(f"Average payoff (S_C, S_C): row={2}, col={2}")
print(f"Average payoff (S_D, S_C): row={3}, col={-7}")
print(f"Average payoff (S_C, S_D): row={-2}, col={6}")
```

```
Stage Nash equilibria:
(array([1., 0.]), array([1., 0.]))

delta = 0.3333333333333333
Average payoff (S_D, S_D): row=-1, col=1
Average payoff (S_C, S_C): row=2, col=2
Average payoff (S_D, S_C): row=3, col=-7
Average payoff (S_C, S_D): row=-2, col=6
```

2. Feasible and individually rational payoff space.

The feasible average payoffs are convex combinations of the four corner payoffs $(-1, 1)$, $(3, -7)$, $(-2, 6)$, $(2, 2)$ (the stage outcomes $(r_1 c_1)$, $(r_1 c_2)$, $(r_2 c_1)$, $(r_2 c_2)$).

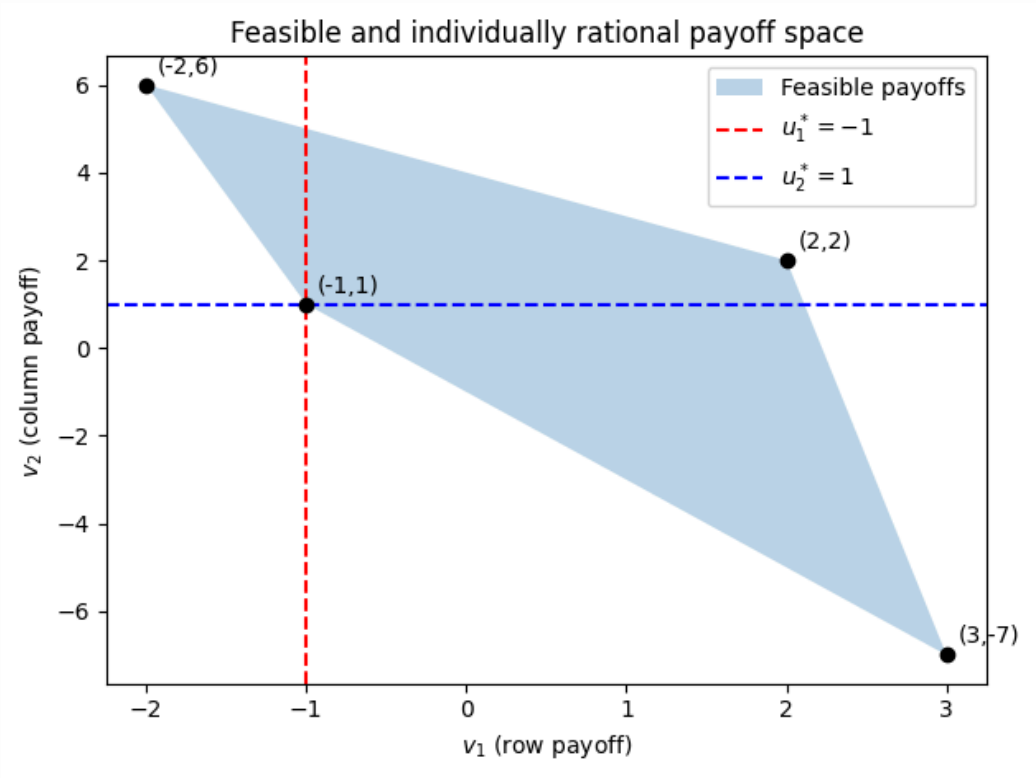
The individually rational payoffs are those feasible payoffs (v_1, v_2) with:

$$v_1 > u_1^* = -1 \qquad \text{and} \qquad v_2 > u_2^* = 1 \tag{7.58}$$

i.e., payoffs that strictly exceed the stage Nash equilibrium payoffs for both players.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.spatial import ConvexHull

# Stage game outcomes: (row payoff, col payoff) for each action pair
outcomes = np.array([[-1, 1], [3, -7], [-2, 6], [2, 2]])

hull = ConvexHull(outcomes)
hull_pts = np.append(hull.vertices, hull.vertices[0])

fig, ax = plt.subplots()
ax.fill(outcomes[hull.vertices, 0], outcomes[hull.vertices, 1],
        alpha=0.3, label="Feasible payoffs")

# Individually rational region: v1 > -1, v2 > 1
# Intersection with feasible region
ax.axvline(-1, color="red", linestyle="--", label="$u_1^* = -1$")
```

```
ax.axhline(1, color="blue", linestyle="--", label="$u_2^* = 1$")

ax.scatter(outcomes[:, 0], outcomes[:, 1], color="black", zorder=5)
for i, (x, y) in enumerate(outcomes):
    ax.annotate(f"({x},{y})", (x, y), textcoords="offset points", xytext=(5,
5))

ax.set_xlabel("$v_1$ (row payoff)")
ax.set_ylabel("$v_2$ (column payoff)")
ax.set_title("Feasible and individually rational payoff space")
ax.legend()
plt.tight_layout()
plt.show()
```



3. Folk Theorem analysis.

By the Folk Theorem, a payoff pair (v_1, v_2) can be sustained as a subgame perfect equilibrium for sufficiently large δ if and only if (v_1, v_2) is:

(a) **Feasible**: a convex combination of the stage game payoff vectors. (b) **Individually rational**: $v_1 > u_1^* = -1$ and $v_2 > u_2^* = 1$.

We check each candidate:

- $(3/2,\ 3/2)$: Is $3/2 > -1$? Yes. Is $3/2 > 1$? Yes. Is $(3/2, 3/2)$ feasible? It lies between $(-1, 1)$, $(2, 2)$, and $(-2, 6)$; it is a convex combination of the corners. **Yes, supportable by the Folk Theorem.**
- $(0,\ 3)$: Is $0 > -1$? Yes. Is $3 > 1$? Yes. Feasibility: $(0, 3)$ lies in the convex hull of the four stage payoffs. **Yes, supportable.**

- $(2, 6)$: Is $2 > -1$? Yes. Is $6 > 1$? Yes. Feasibility: $(2, 6)$ is not one of the corner payoffs. It would require column getting 6 (from r_2, c_1 giving $(-2, 6)$) and row getting 2 simultaneously, which is not a single stage outcome. Check if $(2, 6)$ is in the convex hull: The maximum row payoff when column gets 6 is -2 (from (r_2, c_1)). To get row payoff 2 we need stages (r_2, c_2) giving $(2, 2)$, but then column gets 2 not 6. $(2, 6)$ is **not in the convex hull** of feasible payoffs; it requires both players to receive their maximum simultaneously, which is impossible. **Not supportable.**
- $(2, 0)$: Is $0 > 1$? **No.** This fails individual rationality for the column player. **Not supportable.**

Solution 7.4: Solution to Exercise 4

Consider the Prisoner's Dilemma:

$$M_r = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{7.59}$$

with strategies:

When player 1 using TFT plays against player 2 using Alternator we obtain the following sequence of plays:

1. (C, C) giving utilities: $(3, 3)$
2. (C, D) giving utilities: $(0, 5)$
3. (D, C) giving utilities: $(5, 0)$
4. (C, D) giving utilities: $(0, 5)$
5. (D, C) giving utilities: $(5, 0)$
6. (C, D) giving utilities: $(0, 5)$
7. (D, C) giving utilities: $(5, 0)$
8. ... and so on

This corresponds to:

utilities:

$$\begin{aligned} U_1 &= \left(3 + \sum_{i=1 \text{ when } i \text{ even}}^{\infty} \delta^i 5\right) \\ &= \left(3 + \sum_{i=1}^{\infty} \delta^{2i} 5\right) \\ &= \left(3 + 5\sum_{i=1}^{\infty} \delta^{2i}\right) \\ &= \left(3 + 5\sum_{i=1}^{\infty} \delta^{2^i}\right) \\ &= \left(3 + 5\frac{\delta^2}{1-\delta^2}\right) \end{aligned} \tag{7.60}$$

and equivalently, column player (Alternator) receives 5 on every odd-indexed round (rounds $1, 3, 5, \ldots$ with δ-weights $\delta, \delta^3, \delta^5, \ldots$) plus 3 in round 0:

$$
\begin{aligned}
U_2 &= \left(3 + \sum_{i \ \text{odd} \geq 1}^{\infty} \delta^i \cdot 5\right) \\
&= \left(3 + \sum_{i=0}^{\infty} \delta^{2i+1} \cdot 5\right) \\
&= \left(3 + 5\delta \sum_{i=0}^{\infty} \delta^{2i}\right) \\
&= \left(3 + \frac{5\delta}{1-\delta^2}\right)
\end{aligned}
\tag{7.61}
$$

So the average utilities are:

$$
\begin{aligned}
\bar{U}_1 &= (1-\delta)\left(3 + \frac{5\delta^2}{1-\delta^2}\right) = \frac{3+2\delta^2}{1+\delta} \\
\bar{U}_2 &= (1-\delta)\left(3 + \frac{5\delta}{1-\delta^2}\right) = \frac{3+5\delta-3\delta^2}{1+\delta}
\end{aligned}
\tag{7.62}
$$

When two players playing TFT play each other they both cooperate every round and obtain average payoff 3.

When two players playing Alternator play each other they have utility:

1. (C, C) giving utilities: $(3, 3)$
2. (D, D) giving utilities: $(1, 1)$
3. (C, C) giving utilities: $(3, 3)$
4. (D, D) giving utilities: $(1, 1)$
5. ... and so on

This corresponds to:

$$
\begin{aligned}
U_1 = U_2 &= \left(\sum_{i \ \text{even} \geq 0}^{\infty} \delta^i \cdot 3 + \sum_{i \ \text{odd} \geq 1}^{\infty} \delta^i \cdot 1\right) \\
&= \left(3\sum_{i=0}^{\infty} \delta^{2i} + \sum_{i=0}^{\infty} \delta^{2i+1}\right) \\
&= \frac{3+\delta}{1-\delta^2}
\end{aligned}
\tag{7.63}
$$

with average utility $(1-\delta)(3+\delta)/(1-\delta^2) = (3+\delta)/(1+\delta)$.

This gives the normal-form payoff matrices (rows and columns ordered TFT, Alternator):

$$
M_r = \begin{pmatrix} 3 & \frac{3+2\delta^2}{1+\delta} \\ \frac{3+5\delta-3\delta^2}{1+\delta} & \frac{3+\delta}{1+\delta} \end{pmatrix} \qquad M_c = M_r^T \tag{7.64}
$$

Analysis of Nash equilibria as a function of δ.

TFT (row 1) dominates Alternator (row 2) if and only if it earns weakly more in every column. We check each column in turn.

Column 1 (opponent plays TFT). TFT earns 3; Alternator earns $(3+5\delta-3\delta^2)/(1+\delta)$. TFT does at least as well when:

$$3 \geq \frac{3+5\delta-3\delta^2}{1+\delta} \Leftrightarrow 3+3\delta \geq 3+5\delta-3\delta^2 \Leftrightarrow 3\delta^2 \geq 2\delta \Leftrightarrow \delta \geq \frac{2}{3} \tag{7.65}$$

Column 2 (opponent plays Alternator). TFT earns $(3+2\delta^2)/(1+\delta)$; Alternator earns $(3+\delta)/(1+\delta)$. TFT does at least as well when:

$$\frac{3+2\delta^2}{1+\delta} \geq \frac{3+\delta}{1+\delta} \Leftrightarrow 2\delta^2 \geq \delta \Leftrightarrow \delta \geq \frac{1}{2} \tag{7.66}$$

Since $2/3 > 1/2$, the binding constraint is the first. Therefore:

- $\boldsymbol{\delta > 2/3}$: TFT strictly dominates Alternator. The unique Nash equilibrium is **(TFT, TFT)**, with payoffs $(3, 3)$.
- $\boldsymbol{\delta = 2/3}$: TFT weakly dominates; **(TFT, TFT)** remains a Nash equilibrium.
- $\boldsymbol{1/2 < \delta < 2/3}$: TFT beats Alternator against an Alternator opponent ($b > d$) but Alternator beats TFT against a TFT opponent ($c > a$). Neither pure strategy profile is a Nash equilibrium in this range; a mixed equilibrium exists.
- $\boldsymbol{\delta \leq 1/2}$: Alternator weakly dominates TFT. The unique Nash equilibrium is **(Alternator, Alternator)**, with payoffs $(3+\delta)/(1+\delta) < 3$.

Interpretation. Patient players ($\delta > 2/3$) prefer TFT because the future cooperative gains outweigh the short-term benefit of alternating defection. When δ is low, players discount the future heavily and Alternator's exploitation of TFT in early rounds is attractive. The threshold $\delta^* = 2/3$ marks where sustained cooperation via TFT becomes self-enforcing.

8. Direct Reciprocity

Sustained cooperation over time depends on players remembering past interactions and responding to them appropriately. This chapter begins with Axelrod's computer tournaments, which identified strategies that succeed in the iterated Prisoner's Dilemma, before focusing on reactive strategies: a tractable class that conditions only on the opponent's last move.

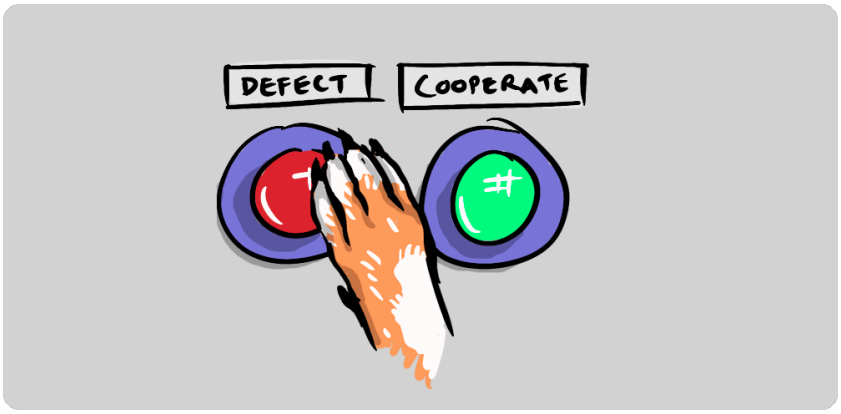


Figure 8.1: Cooperate or defect: in a repeated encounter each player presses one button at a time, with an eye on how the other has played before. Direct reciprocity studies how such conditional responses sustain cooperation.

8.1 Motivating Example: Research Collaboration

Alice and Bob are researchers who meet regularly to share results before submitting papers. At each meeting, each can choose to:

- **Cooperate (C)**: share their latest findings openly.
- **Defect (D)**: withhold key results to protect their competitive advantage.

The payoff structure is that of a **Prisoner's Dilemma**: mutual sharing is best collectively, but each individual has a short-term incentive to withhold. The Folk Theorem tells us that cooperation *can* be sustained in an infinitely repeated game: but it does not say *how*.

In practice, Alice and Bob remember only the last meeting. Alice might decide: "I'll share if you shared last time, but withhold if you withheld." This **memory-one** approach, conditioning solely on the previous round's outcome, is both cognitively realistic and mathematically tractable.

This chapter asks: which reactive strategies actually sustain cooperation, and what are their long-run payoffs?

8.2 Theory

We start by giving a more general definition of Example: Prisoners' Dilemma. This game captures the fundamental structure of direct reciprocity in a repeated game setting.

8.2.1 Definition: Prisoner's Dilemma

The **Prisoner's Dilemma** is the two-player symmetric game:

$$A = \begin{pmatrix} R & S \\ T & P \end{pmatrix} \qquad B = \begin{pmatrix} R & T \\ S & P \end{pmatrix} \tag{8.1}$$

with the constraints:

$$T > R > P > S \qquad 2R > T + S \tag{8.2}$$

- The first constraint ensures **Defect** strictly dominates **Cooperate**.

- The second ensures mutual cooperation is socially optimal.

8.2.1.1 Example: Simplified Prisoner's Dilemma

Under what conditions is the following game a Prisoner's Dilemma?

$$A = \begin{pmatrix} 1 & -\mu \\ 1+\mu & 0 \end{pmatrix} \qquad B = A^T \tag{8.3}$$

This requires $1 + \mu > 1$ and $0 > -\mu$, both of which hold for $\mu > 0$. The constraint $2R > T + S$ gives $2 > 1$, which always holds. This parametrisation is convenient: $\mu > 0$ is the only condition needed.

8.2.2 Axelrod's Tournaments

In 1980, Robert Axelrod organised a computer tournament for the iterated Prisoner's Dilemma (Axelrod, 1980).

- Fourteen strategies were submitted.
- The tournament was a round-robin of 200 iterations, plus a strategy playing uniformly at random.
- Some entries were highly sophisticated; one used a χ^2 test to detect random opponents.
- The winner was the simplest entry: **Tit For Tat** (TFT), which cooperates on the first move then copies the opponent's previous action.

A second tournament with 62 submissions followed (Axelrod, 1980). With participants now aware of TFT's success, many strategies were designed specifically to counter it. TFT won again.

These results suggested four principles for successful cooperation:

- **Don't be envious**: don't strive for a higher score than your opponent.
- **Be nice**: never defect first.
- **Reciprocate**: match cooperation with cooperation, defection with defection.
- **Don't be too clever**: avoid overly complex exploitation.

> **Important**
>
> While influential, these principles have since been challenged by modern research (see Notable Research).

8.2.3 Definition: Reactive Strategy

A **reactive strategy** in the Prisoner's Dilemma is a pair $(p, q) \in [0, 1]^2$ where:

- p is the probability of cooperating given the opponent **cooperated** last round.
- q is the probability of cooperating given the opponent **defected** last round.

The first move is typically set separately (often cooperate), since there is no prior history.

8.2.3.1 Example: Common Reactive Strategies

Strategy	(p, q)	Description
Always Cooperate (AllC)	$(1, 1)$	Cooperates regardless of opponent's action
Always Defect (AllD)	$(0, 0)$	Defects regardless of opponent's action

Strategy	(p, q)	Description
Tit For Tat (TFT)	$(1, 0)$	Copies opponent's last move exactly
Generous TFT (GTFT)	$(1, \varepsilon)$	TFT but occasionally forgives a defection
Random	$(1/2, 1/2)$	Cooperates with probability $1/2$ regardless of context
Generous Reciprocator	$(0.9, 0.3)$	Strongly reciprocates cooperation; sometimes forgives
Suspicious Reciprocator	$(0.7, 0.1)$	Cautiously cooperative; rarely forgives defection

Note

Reactive strategies are a subclass of **memory-one** strategies, which condition on the full previous outcome $(a_1, a_2) \in \{C, D\}^2$ rather than just the opponent's last action. The zero-determinant strategies discussed in Notable Research are memory-one but not reactive.

8.2.4 Definition: Markov Chain of Two Reactive Strategies

When players with reactive strategies (p, q) and (p', q') play an infinitely repeated Prisoner's Dilemma, the pair of actions at each round defines a **Markov chain** with state space $\mathcal{S} = \{CC, CD, DC, DD\}$.

The transition matrix $P \in \mathbb{R}^{4\times 4}$, with rows and columns ordered as (CC, CD, DC, DD), is:

$$M = \begin{pmatrix} pp' & p(1-p') & (1-p)p' & (1-p)(1-p') \\ qp' & q(1-p') & (1-q)p' & (1-q)(1-p') \\ pq' & p(1-q') & (1-p)q' & (1-p)(1-q') \\ qq' & q(1-q') & (1-q)q' & (1-q)(1-q') \end{pmatrix} \tag{8.4}$$

where rows correspond to the current state and columns to the next state.

The entry $M_{ss'}$ is the probability of transitioning from state s to state s'. For example, from state CD (player 1 cooperated, player 2 defected): player 1 now cooperates with probability q (their opponent defected), while player 2 now cooperates with probability p' (their opponent cooperated).

8.2.5 Theorem: Long-Run Average Payoffs via Stationary Distribution

If the Markov chain defined by (p, q) and (p', q') is **ergodic** (see Appendix A3), it has a unique stationary distribution $\pi = (\pi_{CC}, \pi_{CD}, \pi_{DC}, \pi_{DD})$ satisfying:

$$\pi M = \pi \qquad \sum_{s\in\mathcal{S}} \pi_s = 1 \tag{8.5}$$

The long-run average payoffs are then:

$$\bar{u}_1 = R\,\pi_{CC} + S\,\pi_{CD} + T\,\pi_{DC} + P\,\pi_{DD} \tag{8.6}$$

$$\bar{u}_2 = R\,\pi_{CC} + T\,\pi_{CD} + S\,\pi_{DC} + P\,\pi_{DD} \tag{8.7}$$

Note

The chain is ergodic whenever $0 < p, q, p', q' < 1$ (all probabilities strictly interior). For boundary cases such as pure TFT $(p = 1, q = 0)$, the chain may not be ergodic in general; the stationary distribution must be verified separately.

8.2.6 Theorem: Steady-State Distribution for Two Reactive Strategies

When both players use strictly non-deterministic reactive strategies ($0 < p, q, p', q' < 1$), the stationary distribution has a closed form. Define:

$$r_1 = p - q \qquad r_2 = p' - q' \tag{8.8}$$

and:

$$s_1 = \frac{q' r_1 + q}{1 - r_1 r_2} \qquad s_2 = \frac{q r_2 + q'}{1 - r_1 r_2} \tag{8.9}$$

Then the unique stationary distribution is:

$$\pi = \left(s_1 s_2,\, s_1(1 - s_2),\, (1 - s_1) s_2,\, (1 - s_1)(1 - s_2) \right) \tag{8.10}$$

and the long-run average payoffs are:

$$\bar{u}_1 = R\, s_1 s_2 + S\, s_1(1 - s_2) + T\, (1 - s_1) s_2 + P\, (1 - s_1)(1 - s_2) \tag{8.11}$$

$$\bar{u}_2 = R\, s_1 s_2 + T\, s_1(1 - s_2) + S\, (1 - s_1) s_2 + P\, (1 - s_1)(1 - s_2) \tag{8.12}$$

The values s_1 and s_2 are the long-run cooperation probabilities of each player. The stationary distribution factorises as a product of marginals, so the players' long-run actions are independent despite their strategies being correlated round-to-round.

8.2.6.1 Proof

Verify by direct substitution: set $\pi = (s_1 s_2, s_1(1 - s_2), (1 - s_1) s_2, (1 - s_1)(1 - s_2))$ and confirm $\pi M = \pi$ holds with the transition matrix from the definition above:

$$\pi M = (s_1 s_2, s_1(1 - s_2), (1 - s_1) s_2, (1 - s_1)(1 - s_2)) P \tag{8.13}$$

After carrying out the multiplication, each component of πM reduces to the corresponding component of π when s_1 and s_2 satisfy the given expressions. The algebra is routine but lengthy.

8.2.6.2 Example: Long-run payoffs for two reactive strategies

Consider the contractor Prisoner's Dilemma with $R = 3, S = 0, T = 5, P = 1$, and two reactive strategies:

- Player 1 (Generous Reciprocator): $(p, q) = (0.9,\, 0.3)$
- Player 2 (Suspicious Reciprocator): $(p', q') = (0.7,\, 0.1)$

From state CC: player 1 cooperates with $p = 0.9$, player 2 with $p' = 0.7$:

$$CC \to CC : 0.63, \quad CC \to CD : 0.27, \quad CC \to DC : 0.07, \quad CC \to DD : 0.03 \tag{8.14}$$

From state CD: player 1 cooperates with $q = 0.3$, player 2 with $p' = 0.7$:

$$CD \to CC : 0.21, \quad CD \to CD : 0.09, \quad CD \to DC : 0.49, \quad CD \to DD : 0.21 \tag{8.15}$$

From state DC: player 1 cooperates with $p = 0.9$, player 2 with $q' = 0.1$:

$$DC \to CC : 0.09, \quad DC \to CD : 0.81, \quad DC \to DC : 0.01, \quad DC \to DD : 0.09 \tag{8.16}$$

From state DD: player 1 cooperates with $q = 0.3$, player 2 with $q' = 0.1$:

$$DD \to CC : 0.03, \quad DD \to CD : 0.27, \quad DD \to DC : 0.07, \quad DD \to DD : 0.63 \tag{8.17}$$

So the full transition matrix is:

$$M = \begin{pmatrix} 0.63 & 0.27 & 0.07 & 0.03 \\ 0.21 & 0.09 & 0.49 & 0.21 \\ 0.09 & 0.81 & 0.01 & 0.09 \\ 0.03 & 0.27 & 0.07 & 0.63 \end{pmatrix} \tag{8.18}$$

Since all four parameters are strictly interior, we apply the closed-form theorem. With $r_1 = 0.9 - 0.3 = 0.6$ and $r_2 = 0.7 - 0.1 = 0.6$:

$$s_1 = \frac{0.1 \times 0.6 + 0.3}{1 - 0.6 \times 0.6} = \frac{0.36}{0.64} = \frac{9}{16} \qquad s_2 = \frac{0.3 \times 0.6 + 0.1}{0.64} = \frac{0.28}{0.64} = \frac{7}{16} \tag{8.19}$$

The stationary distribution is:

$$\pi = \left(\frac{9}{16} \cdot \frac{7}{16}, \frac{9}{16} \cdot \frac{9}{16}, \frac{7}{16} \cdot \frac{7}{16}, \frac{7}{16} \cdot \frac{9}{16} \right) = \left(\frac{63}{256}, \frac{81}{256}, \frac{49}{256}, \frac{63}{256} \right) \tag{8.20}$$

Long-run average payoffs:

$$\bar{u}_1 = \frac{3 \cdot 63 + 0 \cdot 81 + 5 \cdot 49 + 1 \cdot 63}{256} = \frac{497}{256} \approx 1.94 \tag{8.21}$$

$$\bar{u}_2 = \frac{3 \cdot 63 + 5 \cdot 81 + 0 \cdot 49 + 1 \cdot 63}{256} = \frac{657}{256} \approx 2.57 \tag{8.22}$$

The Suspicious Reciprocator earns a higher long-run payoff. Despite cooperating less, their lower generosity extracts value from the Generous Reciprocator's willingness to forgive defections.

8.3 Exercises

Exercise 8.1:

Justify whether or not the following games are instances of the Prisoner's Dilemma.

Game 1

$$A = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{8.23}$$

Game 2

$$A = \begin{pmatrix} 1 & -1 \\ 2 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix} \tag{8.24}$$

Game 3

$$A = \begin{pmatrix} 1 & -1 \\ 2 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} \tag{8.25}$$

Game 4

$$A = \begin{pmatrix} 6 & 0 \\ 12 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 6 & 12 \\ 0 & 0 \end{pmatrix} \tag{8.26}$$

Exercise 8.2:

Consider the Prisoner's Dilemma with $R = 3, S = 0, T = 5, P = 1$.

1. Write down the transition matrix P for the Markov chain formed by Tit For Tat ($p = 1, q = 0$) playing against Always Cooperate ($p' = 1, q' = 1$).
2. Write down the transition matrix P for the Markov chain formed by Tit For Tat ($p = 1, q = 0$) playing against Always Defect ($p' = 0, q' = 0$).
3. For each case, identify whether the chain is ergodic and, if so, find the stationary distribution and the long-run average payoffs.

Exercise 8.3:

For each of the following pairs of reactive strategies:

1. $p = (1/2, 1/2)$ $\quad$ $p' = (1/2, 1/2)$
2. $p = (1/4, 1/2)$ $\quad$ $p' = (1/2, 1/4)$
3. $p = (1/3, 1/3)$ $\quad$ $p' = (2/3, 1/4)$

Obtain the Markov chain representation for a match and the utilities for both players as a function of R, S, T, P.

Exercise 8.4:

Assuming $p = (x, 1/2)$ and using $R = 3, S = 0, T = 5, P = 1$, find the optimal x against the following players:

1. $p' = (1, 0)$
2. $p' = (1/2, 1/2)$

Interpret these results.

Exercise 8.5:

Consider the Prisoner's Dilemma with $R = 3$, $S = 0$, $T = 5$, $P = 1$ and the four reactive strategies:

- **AllC**: $(p, q) = (1, 1)$, always cooperate.
- **TFT**: $(p, q) = (1, 0)$, cooperate first, then copy the opponent's last move.
- **AllD**: $(p, q) = (0, 0)$, always defect.
- **SR** (Suspicious Reciprocator): $(p, q) = (7/10, 1/10)$.

Assume both players cooperate on the first move.

1. For each of the ten distinct pairings (every strategy against each of the four, including itself, using symmetry to halve the work), identify the long-run stationary distribution and the average payoffs $(\bar{u}_1, \bar{u}_2)$.
2. Write the 4×4 payoff matrix M_r for the repeated game in which AllC, TFT, AllD, and SR are the available actions.
3. Find all pure Nash equilibria of this game and compare the payoffs they deliver.

8.4 Programming

8.4.1 Computing the Stationary Distribution of a Reactive Strategy Pair

Given the transition matrix P for two reactive strategies, the stationary distribution satisfies $\pi P = \pi$. We can find it by solving the linear system $(P^T - I)\pi^T = 0$ subject to $\sum \pi_i = 1$.

```
import numpy as np

def reactive_transition_matrix(p, q, p_prime, q_prime):
    """
    Build the 4x4 transition matrix for reactive strategies (p,q) and (p',q').
    States are ordered: CC, CD, DC, DD.
    """
    M = np.array([
        [p * p_prime,       p * (1 - p_prime),       (1 - p) * p_prime,       (1
- p) * (1 - p_prime)],
        [q * p_prime,       q * (1 - p_prime),       (1 - q) * p_prime,       (1
- q) * (1 - p_prime)],
        [p * q_prime,       p * (1 - q_prime),       (1 - p) * q_prime,       (1
- p) * (1 - q_prime)],
        [q * q_prime,       q * (1 - q_prime),       (1 - q) * q_prime,       (1
- q) * (1 - q_prime)],
    ])
    return M

def stationary_distribution(M):
    """
    Compute the stationary distribution of a transition matrix M.
    """
    n = M.shape[0]
    A = (M.T - np.eye(n))
    A[-1, :] = 1
    b = np.zeros(n)
    b[-1] = 1
    return np.linalg.solve(A, b)

# Generous Reciprocator vs Suspicious Reciprocator
p, q = 0.9, 0.3
p_prime, q_prime = 0.7, 0.1

M = reactive_transition_matrix(p, q, p_prime, q_prime)
print("Transition matrix:")
print(np.round(M, 4))

pi = stationary_distribution(M)
print("\nStationary distribution (CC, CD, DC, DD):")
print(np.round(pi, 4))
```

```
Transition matrix:
[[0.63 0.27 0.07 0.03]
 [0.21 0.09 0.49 0.21]
 [0.09 0.81 0.01 0.09]
 [0.03 0.27 0.07 0.63]]
```

```
Stationary distribution (CC, CD, DC, DD):
[0.2461 0.3164 0.1914 0.2461]
```

```python
# Compute long-run average payoffs
R, S, T, P = 3, 0, 5, 1

u1_bar = R * pi[0] + S * pi[1] + T * pi[2] + P * pi[3]
u2_bar = R * pi[0] + T * pi[1] + S * pi[2] + P * pi[3]

print(f"Long-run average payoff for Player 1: {u1_bar:.4f}")
print(f"Long-run average payoff for Player 2: {u2_bar:.4f}")
```

```
Long-run average payoff for Player 1: 1.9414
Long-run average payoff for Player 2: 2.5664
```

8.4.2 Using the Axelrod Library

The `axelrod` Python library (Knight et al., 2016) provides over 240 strategies and tools to run reproducible tournaments, extending Axelrod's original experiments to much larger and more diverse strategic populations.

8.4.2.1 Studying Reactive Strategies

The Axelrod library provides `axl.ReactivePlayer`, which takes the reactive strategy parameters (p, q) directly, matching the notation used throughout this chapter.

The following code runs a long match between the Generous Reciprocator $(p, q) = (0.9, 0.3)$ and the Suspicious Reciprocator $(p', q') = (0.7, 0.1)$ from the example above and compares the result to the closed-form payoffs.

```python
import axelrod as axl
import numpy as np

generous = axl.ReactivePlayer(probabilities=(0.9, 0.3))
suspicious = axl.ReactivePlayer(probabilities=(0.7, 0.1))

match = axl.Match([generous, suspicious], turns=10000, seed=0)
match.play()
u1, u2 = match.final_score_per_turn()
print(f"Simulated:   Generous={u1:.4f}, Suspicious={u2:.4f}")
print(f"Theoretical: Generous={497/256:.4f}, Suspicious={657/256:.4f}")
```

```
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/axelrod/strategies/
zero_determinant.py:25: SyntaxWarning: "\m" is an invalid escape sequence. Such
sequences will not work in the future. Did you mean "\\m"? A raw string is also
an option.
  &s_{min} &= -\min\\left( \\frac{T - l}{l - S}, \\frac{l - S}{T - l}\\right) <=
s <= 1
```

```
Simulated:   Generous=1.9477, Suspicious=2.5772
Theoretical: Generous=1.9414, Suspicious=2.5664
```

8.4.2.2 Exploring Wider Populations

Reactive strategies are a subclass of **memory-one strategies**, which condition on the full previous outcome $(a_1, a_2) \in \{C, D\}^2$ rather than just the opponent's last action. The Axelrod library includes many built-in memory-one strategies.

```
import axelrod as axl
import numpy as np

# Some notable memory-one strategies included in the library
players = [
    axl.TitForTat(),       # reactive: (p, q) = (1, 0)
    axl.ZDGTFT2(),         # generous zero-determinant strategy
    axl.ZDExtortion(),     # extortionate zero-determinant strategy
    axl.Cooperator(),      # reactive: (p, q) = (1, 1)
    axl.Defector(),        # reactive: (p, q) = (0, 0)
]

tournament = axl.Tournament(
    players=players, turns=200, repetitions=20, seed=0
)
results = tournament.play(progress_bar=False)
print("Ranking:")
for rank, name in enumerate(results.ranked_names):
    print(f"  {rank + 1}. {name}")
```

```
Ranking:
  1. Defector
  2. ZD-Extortion: 0.2, 0.1, 1
  3. Tit For Tat
  4. ZD-GTFT-2: 0.25, 0.5
  5. Cooperator
```

The Axelrod library documentation includes a tutorial showing how to replicate Axelrod's original 1980 tournament, providing a reproducible starting point for exploring how strategy performance depends on the composition of the population.

8.5 Notable Research

The seminal empirical work on direct reciprocity is due to Robert Axelrod (Axelrod, 1980; 1980), synthesised in his widely cited book (Axelrod, 1984). As described in Section 3, Axelrod's tournaments established Tit For Tat as the canonical cooperative strategy and produced four heuristics for successful play.

Subsequent work has challenged these heuristics. The development of the Axelrod Python library (Knight et al., 2016) enabled systematic and reproducible analysis across a far greater population of strategies. In particular, (Glynatsi et al., 2024) studied 45,600 tournaments across diverse strategic populations and parameter regimes, finding that the original heuristics do not generalise. The revised empirical findings suggest:

- Be a little bit envious.
- Be "nice" in non-noisy environments or when game lengths are longer.
- Reciprocate both cooperation and defection appropriately.
- It is acceptable to be clever.
- Adapt to the environment.

The observation that *being envious* can be beneficial echoes a landmark theoretical result. (Press & Dyson, 2012) introduced **zero-determinant strategies**, memory-one strategies that can unilaterally enforce linear payoff relationships, including extortionate dynamics. This paper was described by *MIT Technology Review* as having set "the world of game theory on fire."

However, (Hilbe et al., 2013) and (Knight et al., 2018) showed that extortionate strategies do not tend to survive under evolutionary dynamics, tempering the initial excitement and reinforcing the importance of adaptability.

8.6 Conclusion

This chapter examined how cooperation can emerge in pairwise repeated interactions. Axelrod's tournaments identified Tit For Tat as the canonical cooperative strategy and produced four heuristics for successful play. These heuristics have since been challenged: modern computational work, enabled by large-scale reproducible tournaments, finds that what succeeds depends heavily on the strategic population and game parameters.

The theoretical core of the chapter is the reactive strategy framework. When two players use reactive strategies, their interaction defines a Markov chain over outcome pairs, and the stationary distribution gives long-run average payoffs. This connects the intuitions from Axelrod's tournaments to a rigorous mathematical foundation.

Table 8.1 summarises the key concepts.

Concept	Description
Prisoner's Dilemma	A symmetric game where Defect dominates but mutual cooperation is socially optimal
Reactive strategy (p, q)	Cooperates with probability p after C, q after D
Tit For Tat	Cooperates first, then copies opponent's last move
Markov chain of strategies	States are outcome pairs; transitions determined by (p, q) and (p', q')
Stationary distribution	Long-run fraction of time spent in each state
Long-run average payoff	Payoff weighted by stationary distribution
Zero-determinant strategies	Memory-one strategies enforcing linear payoff relationships
Axelrod's tournaments	Empirical tournaments establishing TFT's success and four cooperation heuristics

Table 8.1: Summary of key concepts in direct reciprocity

Attention

Even the simplest reactive strategies can sustain cooperation or undermine it. The Markov chain framework gives precise predictions about long-run payoffs, and modern research continues to refine our understanding of which strategies thrive across diverse and evolving populations.

8.7 Solutions

Solution 8.1: Solution to Exercise 1

Game 1:

This is a Prisoners Dilemma: $(R, S, T, P) = (3, 0, 5, 1)$.

Game 2:

This is a Prisoners Dilemma: $(R, S, T, P) = (1, -1, 2, 0)$: $2 > 1 > 0 > -1$ and $2 \times 1 > 2 - 1$.

Game 3:

This is not a Prisoner's Dilemma $A \neq B^T$

Game 4:

This is not a Prisoner's Dilemma $A \neq B^T$

Solution 8.2: Solution to Exercise 2

This is a substitution exercise using the Definition: Markov Chain of Two Reactive Strategies:

1. **TFT $(p = 1, q = 0)$ vs AllC $(p' = 1, q' = 1)$.**

Substituting $p = 1, q = 0, p' = 1, q' = 1$:

$$\begin{aligned} M &= \begin{pmatrix} 1 \cdot 1 & 1 \cdot 0 & 0 \cdot 1 & 0 \cdot 0 \\ 0 \cdot 1 & 0 \cdot 0 & 1 \cdot 1 & 1 \cdot 0 \\ 1 \cdot 1 & 1 \cdot 0 & 0 \cdot 1 & 0 \cdot 0 \\ 0 \cdot 1 & 0 \cdot 0 & 1 \cdot 1 & 1 \cdot 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned} \tag{8.27}$$

This chain has two absorbing components. Starting from CC: always stays in CC. Starting from CD: moves to DC, then back to CC, then stays. Starting from DC: moves to CC. Starting from DD: moves to $DC \to CC$.

Since all states eventually reach CC (an absorbing state), the chain is **not ergodic** (it has an absorbing state rather than a unique stationary distribution with full support). However, it has a unique stationary distribution:

$$\pi = (1, 0, 0, 0) \tag{8.28}$$

since all paths lead to CC and stay there.

The long-run average payoffs is given by the Theorem: Long-Run Average Payoffs via Stationary Distribution

$$\bar{u}_1 = R \cdot 1 + S \cdot 0 + T \cdot 0 + P \cdot 0 = R = 3 \tag{8.29}$$

$$\bar{u}_2 = R \cdot 1 + T \cdot 0 + S \cdot 0 + P \cdot 0 = R = 3 \tag{8.30}$$

Both players cooperate in every round, earning 3 each.

Here is some code to confirm the above calculations using functions written in Computing the Stationary Distribution of a Reactive Strategy Pair

```
M = reactive_transition_matrix(p=1, q=0, p_prime=1, q_prime=1)
print(M)
```

```
[[1 0 0 0]
 [0 0 1 0]
 [1 0 0 0]
 [0 0 1 0]]
```

```
stationary_distribution(M)
```

```
array([ 1., -0., -0.,  0.])
```

2. TFT ($p = 1, q = 0$) vs AllD ($p' = 0, q' = 0$).

Substituting $p = 1, q = 0, p' = 0, q' = 0$:

$$\begin{aligned} M &= \begin{pmatrix} 1\cdot 0 & 1\cdot 1 & 0\cdot 0 & 0\cdot 1 \\ 0\cdot 0 & 0\cdot 1 & 1\cdot 0 & 1\cdot 1 \\ 1\cdot 0 & 1\cdot 1 & 0\cdot 0 & 0\cdot 1 \\ 0\cdot 0 & 0\cdot 1 & 1\cdot 0 & 1\cdot 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{8.31}$$

From any state: $CC \to CD \to DD \to DD \to \cdots$ (DD is absorbing). Starting from DC: $DC \to CD \to DD$.

All paths reach DD, so the unique stationary distribution is:

$$\pi = (0, 0, 0, 1) \tag{8.32}$$

The chain is **not ergodic** (absorbing state at DD), but it has a unique stationary distribution with $\pi_{DD} = 1$.

The long-run average payoffs is given by the Theorem: Long-Run Average Payoffs via Stationary Distribution

$$\bar{u}_1 = R \cdot 0 + S \cdot 0 + T \cdot 0 + P \cdot 1 = P = 1 \tag{8.33}$$

$$\bar{u}_2 = R \cdot 0 + T \cdot 0 + S \cdot 0 + P \cdot 1 = P = 1 \tag{8.34}$$

Both players end up in perpetual mutual defection, each earning 1.

```
M = reactive_transition_matrix(p=1, q=0, p_prime=0, q_prime=0)
print(M)
```

```
[[0 1 0 0]
 [0 0 0 1]
 [0 1 0 0]
 [0 0 0 1]]
```

```
stationary_distribution(M)
```

```
array([-0., -0.,  0.,  1.])
```

Solution 8.3: Solution to Exercise 3

This is a substitution exercise using the Definition: Markov Chain of Two Reactive Strategies and the Theorem: Steady-State Distribution for Two Reactive Strategies:

1. $p = (1/2, 1/2)$ $p' = (1/2, 1/2)$

$$M = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} \tag{8.35}$$

Long-run average payoffs:

$$\bar{u}_1 = \frac{P + R + S + T}{4} \tag{8.36}$$

$$\bar{u}_2 = \frac{P + R + S + T}{4} \tag{8.37}$$

2. $p = (1/4, 1/2)$ $p' = (1/2, 1/4)$

$$M = \begin{pmatrix} 1/8 & 1/8 & 3/8 & 3/8 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/16 & 3/16 & 3/16 & 9/16 \\ 1/8 & 3/8 & 1/8 & 3/8 \end{pmatrix} \tag{8.38}$$

Long-run average payoffs:

$$\bar{u}_1 = \frac{110P + 42R + 77S + 60T}{289} \tag{8.39}$$

$$\bar{u}_2 = \frac{110P + 42R + 60S + 77T}{289} \tag{8.40}$$

3. $p = (1/3, 1/3)$ $p' = (2/3, 1/4)$

$$M = \begin{pmatrix} 2/9 & 1/9 & 4/9 & 2/9 \\ 2/9 & 1/9 & 4/9 & 2/9 \\ 1/12 & 1/4 & 1/6 & 1/2 \\ 1/12 & 1/4 & 1/6 & 1/2 \end{pmatrix} \tag{8.41}$$

$$\bar{u}_1 = \frac{22P + 7R + 11S + 14T}{54} \tag{8.42}$$

$$\bar{u}_2 = \frac{22P + 7R + 14S + 11T}{54} \tag{8.43}$$

Here is some code to verify the above calculations:

```
import sympy as sym

R = sym.Symbol("R")
S = sym.Symbol("S")
T = sym.Symbol("T")
P = sym.Symbol("P")

for (p, q), (p_prime, q_prime) in [
    ([1 / 2, 1 / 2], [1 / 2, 1 / 2]),
    ([1 / 4, 1 / 2], [1 / 2, 1 / 4]),
    ([1 / 3, 1 / 3], [2 / 3, 1 / 4]),
]:
    M = reactive_transition_matrix(p, q, p_prime, q_prime)
    pi = stationary_distribution(M).round(3)
    u1_bar = R * pi[0] + S * pi[1] + T * pi[2] + P * pi[3]
    u2_bar = R * pi[0] + T * pi[1] + S * pi[2] + P * pi[3]
    print("=====")
    print(f"(p, q)={(p, q)}, (p', q')={(p_prime, q_prime)}")
    print("gives:")
    print(M)
    print(
        "With utility:",
        u1_bar,
        u2_bar,
    )
```

```
=====
(p, q)=(0.5, 0.5), (p', q')=(0.5, 0.5)
gives:
[[0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]]
With utility: 0.25*P + 0.25*R + 0.25*S + 0.25*T 0.25*P + 0.25*R + 0.25*S +
0.25*T
=====
(p, q)=(0.25, 0.5), (p', q')=(0.5, 0.25)
gives:
[[0.125  0.125  0.375  0.375 ]
 [0.25   0.25   0.25   0.25  ]
 [0.0625 0.1875 0.1875 0.5625]
 [0.125  0.375  0.125  0.375 ]]
With utility: 0.381*P + 0.145*R + 0.266*S + 0.208*T 0.381*P + 0.145*R +
0.208*S + 0.266*T
=====
(p, q)=(0.3333333333333333, 0.3333333333333333), (p', q')=(0.6666666666666666,
0.25)
gives:
[[0.22222222 0.11111111 0.44444444 0.22222222]
 [0.22222222 0.11111111 0.44444444 0.22222222]
 [0.08333333 0.25       0.16666667 0.5       ]
 [0.08333333 0.25       0.16666667 0.5       ]]
With utility: 0.407*P + 0.13*R + 0.204*S + 0.259*T 0.407*P + 0.13*R + 0.259*S
+ 0.204*T
```

Solution 8.4: Solution to Exercise 4

This is a substitution exercise using the Theorem: Steady-State Distribution for Two Reactive Strategies:

1. $p' = (1, 0)$

$$u(x) = \frac{(-10x + 4(x-1)^2 + 13)}{(2x-3)^2} \tag{8.44}$$

The derivative of this function is given by:

$$\frac{2(6x-7)}{(2x-3)^3} \tag{8.45}$$

This derivative has zero for $x = 7/6$ which is > 1. Thus the utility is monotonic over the interval $[0, 1]$. We have (by substitution):

$$u(0) = 17/9 \qquad u(1) = 3 \tag{8.46}$$

Thus $u(x)$ is an increasing function so the optimal value of x is 1.

Against a player that is unforgiving (reacts to defection with defection), given that our player will play randomly against a defection it is better to always cooperate.

2. $p' = (1/2, 1/2)$

$$u(x) = -3x/4 + 21/8 \tag{8.47}$$

This is a decreasing function so the optimal value of x is 0.

Against a random player (who takes no notice of what we do) it is better to defect.

Here is some code to verify our calculations (using a function that implements Theorem: Steady-State Distribution for Two Reactive Strategies):

```
def theoretic_steady_state(p, q, p_prime, q_prime):

    r_1 = p - q
    r_2 = p_prime - q_prime
    s_1 = (q_prime * r_1 + q) / (1 - r_1 * r_2)
    s_2 = (q * r_2 + q_prime) / (1 - r_1 * r_2)
    return np.array([s_1 * s_2, s_1 * (1 - s_2), (1 - s_1) * s_2, (1 - s_1) *
(1 - s_2)])

R, S, T, P = sym.Symbol("R"), sym.Symbol("S"), sym.Symbol("T"),
sym.Symbol("P"),

def theoretic_utility(p, q, p_prime, q_prime, rstp=np.array([R, S, T, P])):
    pi = theoretic_steady_state(p, q, p_prime, q_prime)
    return np.dot(pi, rstp)

x = sym.Symbol("x")
for (p_prime, q_prime) in [(sym.S(1), sym.S(0)), (sym.S(1) / 2, sym.S(1) /
2)]:
    print(f"(p', q')={(p_prime, q_prime)}")
    utility = sym.factor(theoretic_utility(x, sym.S(1) / 2, p_prime, q_prime,
```

```
rstp=np.array((3, 0, 5, 1))))
    print(f"    u(x)={utility}")
    print(f"    u(0)={utility.subs({x:0})}")
    print(f"    u(1)={utility.subs({x:1})}")
    print(f"    du(x)/dx={utility.diff(x).simplify()}")
    print(f"    solution of du(x)/dx=0: {sym.solveset(utility.diff(x), x)}")
```

```
(p', q')=(1, 0)
    u(x)=(4*x**2 - 18*x + 17)/(2*x - 3)**2
    u(0)=17/9
    u(1)=3
```

```
    du(x)/dx=2*(6*x - 7)/(8*x**3 - 36*x**2 + 54*x - 27)
    solution of du(x)/dx=0: {7/6}
(p', q')=(1/2, 1/2)
    u(x)=-3*(2*x - 7)/8
    u(0)=21/8
    u(1)=15/8
    du(x)/dx=-3/4
    solution of du(x)/dx=0: EmptySet
```

Solution 8.5: Solution to Exercise 5

Part 1.

AllC, TFT, and AllD are at the boundary of the parameter space, so we trace those chains directly. SR has strictly interior parameters, so the closed-form theorem applies whenever SR is one of the players. Since the PD is symmetric, $(\bar{u}_1, \bar{u}_2)$ for A vs B equals $(\bar{u}_2, \bar{u}_1)$ for B vs A, halving the number of computations.

Pairings among AllC, TFT, AllD (traced directly, starting from CC):

- **AllC vs AllC.** Chain stays in CC. $\pi = (1, 0, 0, 0)$. Payoffs $(3, 3)$.
- **AllC vs TFT.** AllC cooperates always; TFT sees cooperation and reciprocates. Chain stays in CC. $\pi = (1, 0, 0, 0)$. Payoffs $(3, 3)$.
- **AllC vs AllD.** AllC cooperates, AllD defects unconditionally. State CD is absorbing. $\pi = (0, 1, 0, 0)$. Payoffs $(0, 5)$.
- **TFT vs TFT.** Starting from CC: both cooperate each round. $\pi = (1, 0, 0, 0)$. Payoffs $(3, 3)$.
- **TFT vs AllD.** $CC \to CD \to DD$; DD absorbing. $\pi = (0, 0, 0, 1)$. Payoffs $(1, 1)$.
- **AllD vs AllD.** $CC \to DD$; DD absorbing. $\pi = (0, 0, 0, 1)$. Payoffs $(1, 1)$.

Pairings involving SR (closed-form theorem, $r_1 = p - q$, $r_2 = p' - q'$):

Write SR $= (7/10, 1/10)$ so $r_{\text{SR}} = 7/10 - 1/10 = 3/5$.

SR vs AllC. $r_1 = 3/5$, $r_2 = 1 - 1 = 0$. Denominator $= 1$.

$$s_1 = \frac{3}{5} \cdot 1 + \frac{1}{10} = \frac{7}{10}, \qquad s_2 = \frac{1}{10} \cdot 0 + 1 = 1 \tag{8.48}$$

$\pi = (7/10, 0, 3/10, 0)$.

$$\bar{u}_1 = 3 \cdot \frac{7}{10} + 5 \cdot \frac{3}{10} = \frac{36}{10} = \frac{18}{5}, \qquad \bar{u}_2 = 3 \cdot \frac{7}{10} = \frac{21}{10} \tag{8.49}$$

SR vs TFT. $r_1 = 3/5$, $r_2 = 1 - 0 = 1$. Denominator $= 1 - 3/5 = 2/5$.

$$s_1 = \frac{0 \cdot \frac{3}{5} + \frac{1}{10}}{\frac{2}{5}} = \frac{\frac{1}{10}}{\frac{2}{5}} = \frac{1}{4}, \qquad s_2 = \frac{\frac{1}{10} \cdot 1 + 0}{\frac{2}{5}} = \frac{1}{4} \tag{8.50}$$

$\pi = (1/16, 3/16, 3/16, 9/16)$.

$$\bar{u}_1 = \frac{3 + 0 + 15 + 9}{16} = \frac{27}{16}, \qquad \bar{u}_2 = \frac{3 + 15 + 0 + 9}{16} = \frac{27}{16} \tag{8.51}$$

SR vs AllD. $r_1 = 3/5$, $r_2 = 0 - 0 = 0$. Denominator $= 1$.

$$s_1 = 0 \cdot \frac{3}{5} + \frac{1}{10} = \frac{1}{10}, \qquad s_2 = \frac{1}{10} \cdot 0 + 0 = 0 \tag{8.52}$$

$\pi = (0, 1/10, 0, 9/10)$.

$$\bar{u}_1 = 0 \cdot \frac{1}{10} + 1 \cdot \frac{9}{10} = \frac{9}{10}, \qquad \bar{u}_2 = 5 \cdot \frac{1}{10} + 1 \cdot \frac{9}{10} = \frac{7}{5} \tag{8.53}$$

SR vs SR. $r_1 = r_2 = 3/5$. Denominator $= 1 - 9/25 = 16/25$.

$$s_1 = \frac{\frac{1}{10} \cdot \frac{3}{5} + \frac{1}{10}}{\frac{16}{25}} = \frac{\frac{8}{50}}{\frac{16}{25}} = \frac{1}{4}, \qquad s_2 = \frac{1}{4} \tag{8.54}$$

$\pi = (1/16, 3/16, 3/16, 9/16)$.

$$\bar{u}_1 = \bar{u}_2 = \frac{27}{16} \tag{8.55}$$

Part 2. Reading off the row player's payoff, with rows and columns ordered (AllC, TFT, AllD, SR) and symmetric pairs filled in by the PD symmetry:

$$M_r = \begin{pmatrix} 3 & 3 & 0 & \frac{21}{10} \\ 3 & 3 & 1 & \frac{27}{16} \\ 5 & 1 & 1 & \frac{7}{5} \\ \frac{18}{5} & \frac{27}{16} & \frac{9}{10} & \frac{27}{16} \end{pmatrix} \tag{8.56}$$

Since the PD is symmetric, $M_c = M_r^T$.

Part 3. We test each pure strategy pair.

(AllC, AllC): Row deviates to AllD: $5 > 3$. **Not a NE.**

(TFT, TFT): Deviating to AllC earns $3 = 3$ (no gain); to AllD earns $1 < 3$; to SR earns $27/16 < 3$. **Is a NE.**

(AllD, AllD): Deviating to AllC earns $0 < 1$; to TFT earns $1 = 1$ (no gain); to SR earns $9/10 < 1$. **Is a NE.**

(SR, SR): SR earns $27/16 \approx 1.69$. Deviating to AllC earns $21/10 = 2.1 > 27/16$ (equivalently $336/160 > 270/160$), so the deviation is profitable; to TFT earns $27/16$ (a tie); to AllD earns $7/5 = 1.4 < 27/16$. Since deviating to AllC pays, this is **not a NE.**

The two pure Nash equilibria are **(TFT, TFT)** and **(AllD, AllD)**. The (TFT, TFT) equilibrium payoff-dominates (AllD, AllD), with both players earning 3 rather than 1.

We can confirm the payoff matrix, and the resulting equilibria, with the `axelrod` library by simulating a long match between each pair of strategies.

```
import axelrod as axl
import numpy as np

# Reactive parameters (p, q); reactive players cooperate on the first move.
strategies = {"AllC": (1, 1), "TFT": (1, 0), "AllD": (0, 0), "SR": (0.7, 0.1)}
order = ["AllC", "TFT", "AllD", "SR"]


def long_run_payoff(row, col, turns=5000, seed=1):
    players = [
        axl.ReactivePlayer(probabilities=strategies[row]),
        axl.ReactivePlayer(probabilities=strategies[col]),
    ]
    match = axl.Match(players, turns=turns, seed=seed)
    match.play()
    return match.final_score_per_turn()[0]


payoff_matrix = np.array(
    [[long_run_payoff(row, col) for col in order] for row in order]
)
print("Row-player payoff matrix (AllC, TFT, AllD, SR):")
print(np.round(payoff_matrix, 3))
```

```
Row-player payoff matrix (AllC, TFT, AllD, SR):
[[3.000e+00 3.000e+00 1.000e-03 2.105e+00]
 [3.000e+00 3.000e+00 1.000e+00 1.642e+00]
 [5.000e+00 1.001e+00 1.000e+00 1.369e+00]
 [3.596e+00 1.643e+00 9.080e-01 1.645e+00]]
```

```
# A profile (s, s) is a pure Nash equilibrium when no row deviation beats it.
for index, strategy in enumerate(order):
    own = payoff_matrix[index, index]
    best_deviation = payoff_matrix[:, index].max()
    print(
        f"({strategy}, {strategy}): payoff {own:.3f}, "
        f"best response {best_deviation:.3f}, "
        f"Nash equilibrium: {own >= best_deviation - 1e-6}"
    )
```

```
(AllC, AllC): payoff 3.000, best response 5.000, Nash equilibrium: False
(TFT, TFT): payoff 3.000, best response 3.000, Nash equilibrium: True
(AllD, AllD): payoff 1.000, best response 1.000, Nash equilibrium: True
(SR, SR): payoff 1.645, best response 2.105, Nash equilibrium: False
```

9. The Biological Origins of Evolutionary Game Theory

Evolutionary biology poses questions that look surprisingly like strategic games: which traits persist in a population, and why? This chapter traces the biological origins of evolutionary game theory and motivates the dynamical models developed in the chapters that follow. It is intended as wider context rather than mathematical machinery: a reader following a purely mathematical path through the book can skip ahead to Replicator Dynamics without loss of continuity.

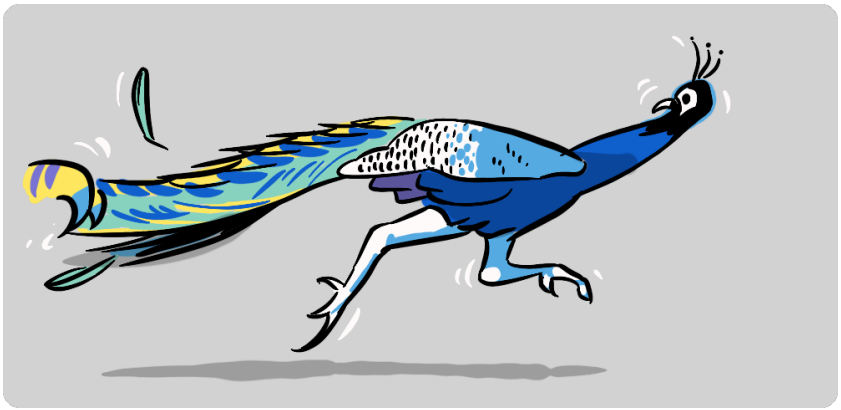


Figure 9.1: A peacock's tail is extravagant and costly to carry, yet it persists. Such apparently wasteful traits are a puzzle that evolutionary game theory helps to explain, much as it does the stag's antlers in the example that follows.

9.1 Motivating Example: Why Don't Stags Have Bigger Antlers?

Every autumn on the Scottish hillside, red deer stags compete for access to females. Stags with larger, heavier antlers are more likely to win contests (Clutton-Brock et al., 1982; Kruuk et al., 2002), but growing such antlers over the summer demands significant energy that could otherwise be allocated to body condition, immune function, and winter survival (Harshman & Zera, 2007; Moen et al., 1999).

Now consider the next few generations, as heavy antler growth spreads. Most stags now have large antlers. Contests between them are no longer quick or risk-free; both stags hold their ground, fights escalate, and serious injury becomes more likely. The energy diverted into growing large antlers no longer provides the same advantage it once did. A stag that allocates slightly less energy to antler growth may survive more winters, have more breeding opportunities, and ultimately achieve higher reproductive success than its heavily armed competitors (Enquist & Leimar, 1987).

Thus, selection favours heavier antlers when they are rare, but this advantage diminishes as they become common. The fitness of any given antler size depends on what the rest of the population is doing: it is a game (Maynard Smith & Price, 1973).

Why the stable antler size is not simply "as large as possible" cannot be answered by classical genetics alone; it requires the framework of evolutionary game theory, developed in this chapter.

9.2 Theory

9.2.1 Darwin's Three Conditions

In 1859 Darwin proposed that the diversity of life could arise from a single, simple mechanism: no designer, no foresight, no goal. Three observable facts about populations are sufficient:

1. **Variation.** Individuals differ from one another in heritable traits: body size, beak shape, defensive behaviour.
2. **Heritability.** Offspring tend to resemble their parents more than they resemble randomly chosen members of the population. Traits are passed down.

3. **Differential reproduction.** Some individuals, by virtue of their traits, survive longer or leave more offspring than others.

When these three conditions hold, the composition of a population changes over time. Traits that help their bearers reproduce become more common in the next generation; traits that hinder reproduction fade away. Darwin called this process **natural selection**.

> **Note**
>
> Natural selection is not a force pushing organisms toward some ideal form. It is a filtering process: variants that reproduce more simply leave more copies of themselves. There is no planning, no goal, no optimisation, only differential survival and reproduction.

All three conditions hold for stag antlers. Antler size varies across individuals. It is partly inherited. And it affects how many offspring a stag leaves (via contest outcomes). Natural selection therefore acts on antler size; the only question is in which direction.

9.2.2 Genes, Loci, and Alleles

Darwin's three conditions require heritable variation but say nothing about how inheritance works. The vocabulary of genetics makes the mechanism precise, and the same vocabulary maps cleanly onto the language of strategies.

A **gene** is a stretch of DNA that carries the instructions for some heritable feature. The fixed position a gene occupies on a chromosome is its **locus**, and the variant forms a gene can take at that locus are its **alleles**. Most animals are **diploid**: they carry two copies of each chromosome, and hence two alleles at every locus, one inherited from each parent. An individual carrying two copies of the same allele is **homozygous** at that locus; one carrying two different alleles is **heterozygous**.

When the two alleles differ, one is often **dominant** and masks the effect of the other, which is then **recessive**. Figure 9.2 shows a single locus with two alleles, the three diploid genotypes they produce, and the way those genotypes map to an observable feature when one allele is dominant.

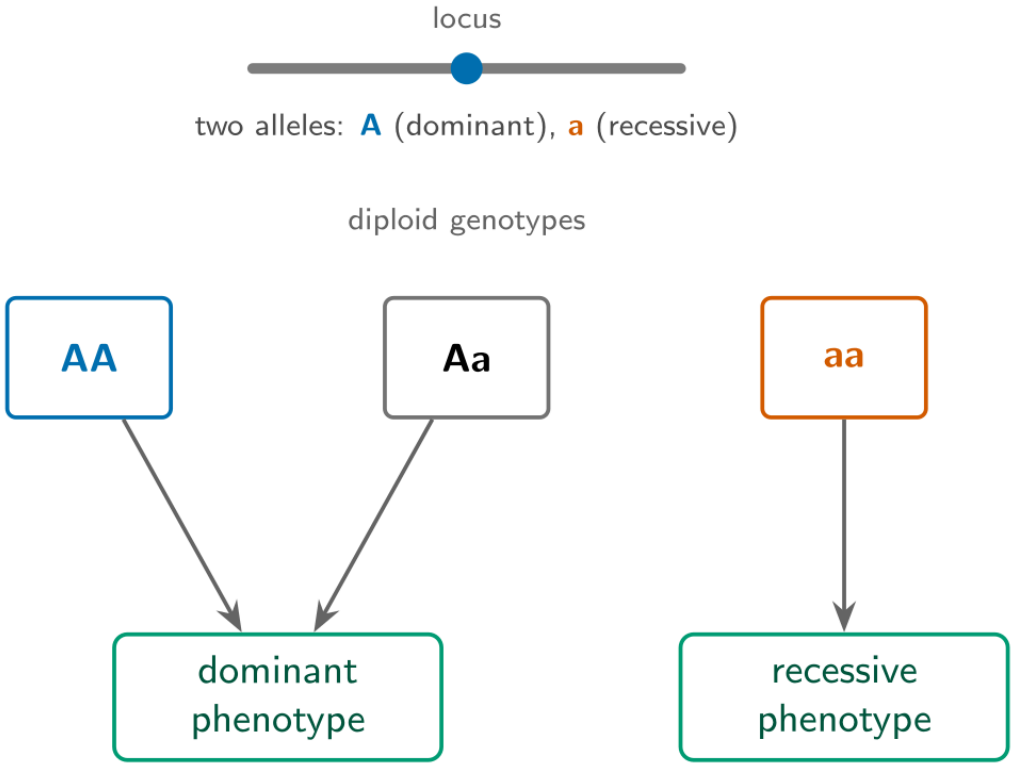


Figure 9.2: A single locus carrying two alleles: the dominant **A** and the recessive **a**. The three diploid genotypes are **AA**, **Aa**, and **aa**. When **A** is dominant, both **AA** and **Aa** produce the dominant phenotype, and only **aa** produces the recessive one.

The central bookkeeping quantity of population genetics is the **allele frequency**: the proportion of all copies at a locus that are of a given allele. At its most basic, natural selection is change in allele frequency from one generation to the next. This is the biological counterpart of the strategy frequency x that the later chapters track.

9.2.3 Genotype and Phenotype

The **genotype** of an organism is its inherited genetic constitution: the alleles it carries across all relevant loci. The genotype is fixed at conception and is what passes, one allele per locus, to each offspring.

The **phenotype** is the observable expression of that genotype in a given environment: the physical structure, physiological properties, and behavioural tendencies visible to natural selection. Antler size, plumage colour, and the propensity to escalate or retreat in a contest are all phenotypic traits.

The map from genotype to phenotype is rarely one-to-one. Most interesting traits are **polygenic**, shaped by many loci acting together, and the same genotype can yield different phenotypes in different environments. A stag well fed over the summer grows larger antlers than a genetically identical stag on poorer ground.

The crucial asymmetry is this: natural selection acts on phenotypes but changes the frequency of genotypes. A stag's genotype specifies its antler-investment programme; the antlers that result are the phenotype; contest outcomes determine fitness; and fitness determines how many copies of the underlying genotype reach the next generation. Figure 9.3 shows the loop.

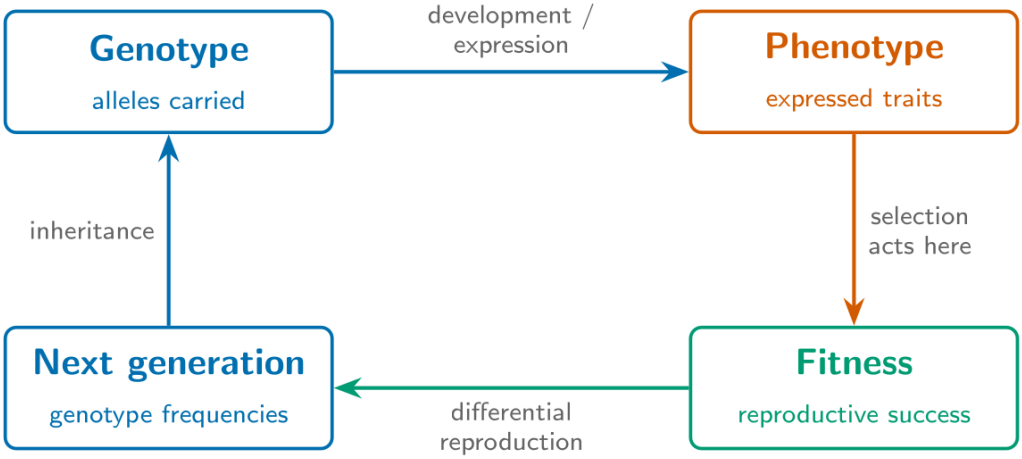


Figure 9.3: The cycle of selection. The genotype develops into a phenotype; selection acts on the phenotype through reproductive success; differential reproduction changes the genotype frequencies in the next generation; and inheritance carries those genotypes forward. Selection reads the phenotype but rewrites the genotype frequencies.

The game-theoretic translation is direct: each distinct genotype (or allele at the relevant locus) corresponds to a **strategy**, and the phenotype is the strategy in action. What the replicator equation tracks over time is the frequency of genotypes, that is, of strategies, in the population.

9.2.4 Pleiotropy: One Gene, Many Traits

A single gene rarely affects just one trait. **Pleiotropy** is the phenomenon of one gene influencing several apparently unrelated phenotypic traits at once. Figure 9.4 illustrates the idea: a single allele feeds into more than one trait, so selection acting on one of those traits inevitably drags the others along with it.

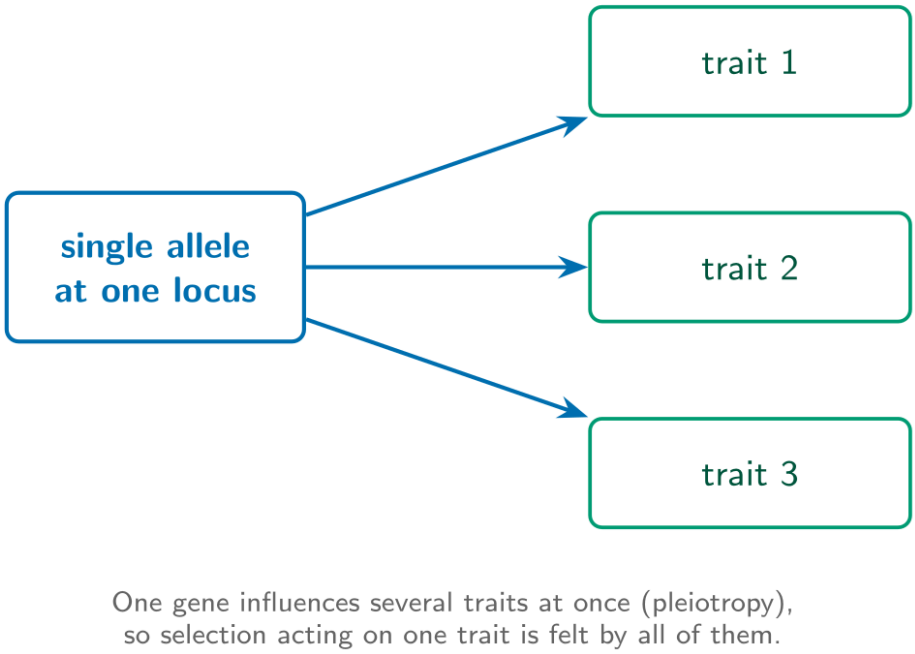


Figure 9.4: Pleiotropy. A single allele at one locus influences several traits at once, so the traits cannot evolve independently of one another.

Pleiotropy matters here because it explains why a trait that lowers fitness on one axis need not be eliminated. An allele can be costly in one respect and beneficial in another, and selection acts on the net effect. This is a recurring theme: a strategy that looks individually disadvantageous can persist because of how it is tied to everything else going on in the population.

9.2.5 What Is a "Strategy" in Biology?

In game theory a strategy is a decision rule. In biology a **strategy** is an inherited behavioural tendency: the tendency to fight or retreat, to cooperate or defect, to invest heavily in display or conserve resources.

Biological strategies are not chosen. A stag does not calculate expected payoffs and decide on an antler investment level. It expresses a phenotype inherited from its parents. Individuals that happen to inherit strategies effective against the opponents they actually encounter leave more offspring. Those offspring inherit the same tendencies. Over generations, effective strategies spread.

The word "strategy" in evolutionary biology is shorthand for an *inherited behavioural phenotype*. The rational-agent interpretation is a useful fiction that lets us apply game-theoretic mathematics to a process that is entirely mechanical.

9.2.6 Fitness: Reproductive Success as Payoff

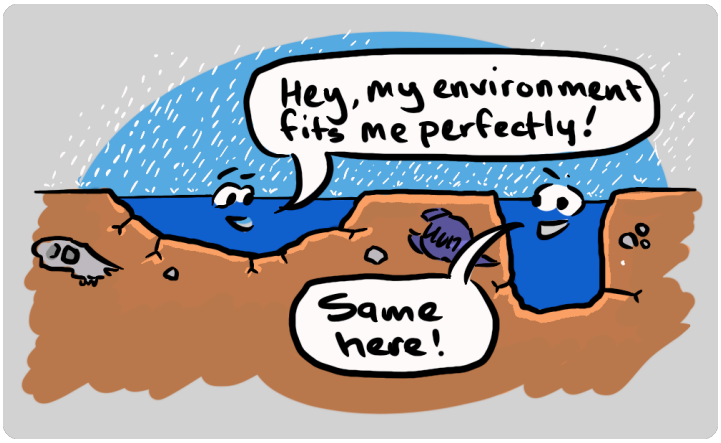


Figure 9.5: "Survival of the fittest" is often misread as a contest, when it is really a statement of fit. A puddle is not in competition with the hole it sits in: it simply takes the shape that the hole allows. Fitness works the same way, it is not how strong or clever a type is in the abstract, but how well its traits match the environment it happens to occupy.

The central quantity in evolutionary biology is **fitness**, the expected number of surviving offspring an individual leaves. Variants with higher fitness leave more copies of themselves and so increase in frequency; variants with lower fitness decline. Fitness is the biological counterpart of a payoff: it is the currency in which natural selection keeps score.

In classical population genetics, fitness is treated as a fixed property of a type, a constant attached to each genotype regardless of how common it is. A type with above-average fitness then grows in frequency, generation on generation, until it dominates the population.

The stag example does not fit this picture. The advantage of heavy antler investment is large when such investment is rare and small when it is common, so fitness is not a fixed constant: it depends on the current composition of the population. Making that dependence precise is the subject of the next section, and turning it into a dynamical equation is the work of Replicator Dynamics.

9.2.7 Frequency-Dependent Selection

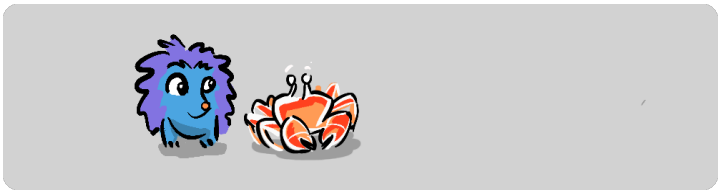


Figure 9.6: Cooperation between very different creatures. Whether such behaviour pays off depends on how common it is in the population, an idea made precise by frequency-dependent selection.

Selection is **frequency-dependent** when the fitness of a type depends on the current composition of the population, not just on its own properties in isolation.

Frequency dependence arises whenever individuals interact, which is essentially always in real populations. The stag antler example is a canonical instance, but the phenomenon is ubiquitous:

- A bacterium that secretes a costly public-good molecule benefits the group but is exploited by non-secretors. Its fitness advantage depends on how many non-secretors are present.
- A fish that mimics a toxic species is protected by predators, but only while the toxic species remains common enough for predators to have learned the association.
- A worker bee that reproduces rather than works gains personally in the short run, but a colony of non-workers produces nothing and dies.

In all these cases, what makes a strategy "good" depends on what the rest of the population is doing.

A classical example of frequency-dependent selection predates Maynard Smith and Price by decades. Fisher's argument for the 1:1 sex ratio (Fisher, 1930) is a frequency-dependent argument in disguise: if either sex becomes rare, the per-capita reproductive success of producing offspring of that sex rises, so parents who invest in the rarer sex leave more grandchildren. The stable investment ratio is the one at which neither sex offers an advantage. This is the same balance condition that determines x^* in Hawk–Dove and, more broadly, every mixed equilibrium we meet in the chapters that follow.

9.2.8 The Hawk–Dove Game

The stag example is formalised by the **Hawk–Dove game**, introduced by Maynard Smith and Price (Maynard Smith & Price, 1973) to explain why animal contests so rarely escalate to serious injury.

Two strategies:

- **Hawk** (H): always escalate; fight until injured or opponent retreats.
- **Dove** (D): display first; retreat immediately if opponent escalates.

Each contest is over a resource worth $V > 0$ to the winner. A fight between two Hawks injures one of them at cost C, where $C > V$. The expected payoffs for each encounter type are:

$$\begin{array}{lcc} & \text{meets H} & \text{meets D} \\ \text{H plays} & \frac{V-C}{2} & V \\ \text{D plays} & 0 & \frac{V}{2} \end{array} \tag{9.1}$$

Each strategy's expected fitness depends on how common Hawks are. Writing x for the fraction of Hawks, a Hawk does worse as x rises, since it more often meets another Hawk and risks a costly fight, while a Dove does best when Hawks are rare. The two fitnesses are equal at a single interior frequency,

$$x^* = \frac{V}{C}, \tag{9.2}$$

which lies strictly between 0 and 1 because $C > V$. The stable outcome is therefore a **mixed population**, not fixation of either pure strategy: when Hawks are rarer than x^* they do better and spread, and when they are commoner they do worse and decline, so the population is driven back to x^* from either side, as Figure 9.7 shows. This balance point is a Nash equilibrium of the underlying game. The fitnesses are derived in full, and the sense in which x^* is evolutionarily stable is made precise, in Replicator Dynamics.

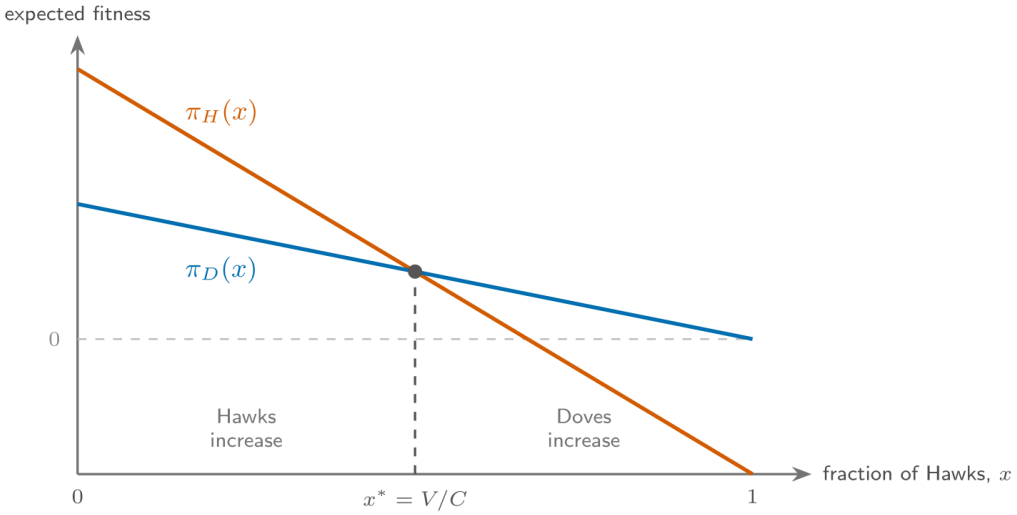


Figure 9.7: Frequency-dependent fitness in the Hawk–Dove game. Each line gives the expected fitness of a strategy as the fraction of Hawks x varies. The lines cross at $x^* = V/C$: to the left of x^* Hawks earn more and increase in frequency, to the right Doves earn more and Hawks decline. The population is driven back to x^* from either side.

Note

In the stag example, "Hawk" corresponds to heavy antler investment and "Dove" to light investment. The stable antler investment level in the population is $x^* = V/C$, determined entirely by the ratio of the resource value to the cost of escalation. Real stag populations show exactly this kind of stable polymorphism in investment strategies.

Two biological terms are worth noting at this point. A population is **monomorphic** if every individual plays the same strategy, and **polymorphic** if more than one strategy is present. The Hawk–Dove equilibrium admits both readings: as a polymorphic population in which a fraction V/C of individuals are pure Hawks and the rest pure Doves, or as a monomorphic population in which every individual plays the mixed strategy $\sigma^* = (V/C, 1 - V/C)$. The two readings give the same expected payoffs in a well-mixed population, and the mathematical chapters that follow treat them interchangeably.

9.2.9 Evolutionary stability

The Hawk–Dove game shows that neither pure Hawk nor pure Dove takes over: at $x^* = V/C$ each type has the same expected fitness. A sharper question is whether this composition is itself stable against the entry of novel behaviour. If a rare mutant playing a different strategy appeared, would selection eliminate it, or would it spread?

The mixed composition at x^* has the property that residents always do strictly better than any rare mutant, so the mutant fades and the population returns to x^*. Strategies with this property are called **evolutionarily stable**, a notion introduced by Maynard Smith and Price (Maynard Smith & Price,

1973). For the Hawk–Dove game with $V < C$, neither pure Hawk nor pure Dove is evolutionarily stable, but the mixed composition $\sigma^* = (V/C, 1 - V/C)$ is. The formal definition and its algebraic characterisation are developed in Replicator Dynamics.

9.2.10 The Bridge: Fitness Is Payoff

The conceptual step from population genetics to evolutionary game theory requires a single substitution:

> **Replace the fixed fitness constants of classical genetics with payoffs that depend on what strategies you encounter.**

In a pairwise interaction model, where each individual is paired at random with another drawn from the population, the expected fitness of an individual using strategy i in a population where strategy j has frequency x_j is:

$$\pi_i(x) = \sum_j x_j \cdot M_{ij} \tag{9.3}$$

where M_{ij} is the payoff to strategy i when meeting strategy j. The payoff matrix M is the fitness matrix. Strategies with above-average fitness increase in frequency; strategies with below-average fitness decline.

Population genetics	Evolutionary game theory
Genotype / allele	Strategy
Absolute fitness W_i	Expected payoff π_i
Fitness advantage over mean	$\pi_i - \bar{\pi}$ drives frequency change
Frequency-dependent fitness	Payoff matrix M_{ij}
Mutation rate μ	Strategy exploration / trembling hand
Population size N	Governs how much chance matters vs selection
Genetic drift	Stochastic fluctuations in finite-population models
Evolutionarily stable strategy	Nash equilibrium that selection cannot destabilise

Table 9.1: The correspondence between population genetics and evolutionary game theory

Note

Two rows of Table 9.1 refer to concepts developed in later chapters. **Mutation** is a random, heritable change in the genotype: a copying error during reproduction that can introduce a new allele, and hence a new strategy, into the population. In the replicator-mutator equation, mutation is modelled as a small probability that offspring adopt a randomly drawn strategy rather than the parent's. **Genetic drift** is the random fluctuation of allele frequencies that arises from finite population size: even a fitter type can be eliminated by bad luck in a small enough population. Drift is negligible in the large-population limit governed by the replicator equation, but it is central to the Moran Process, which models evolution one birth-death event at a time.

9.2.11 What the Following Chapters Formalise

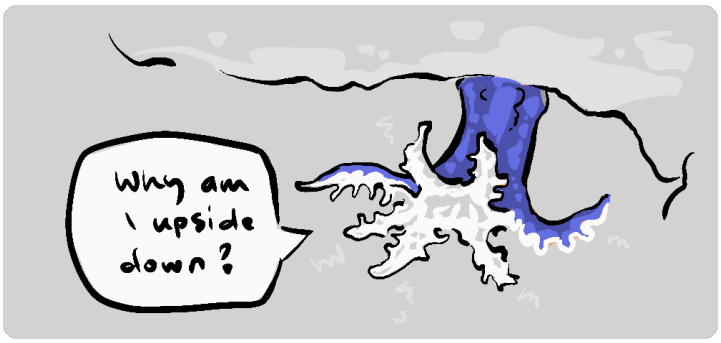


Figure 9.8: A single polyp is a simple creature, yet colonies of them build reefs. Much of what follows asks how the behaviour of many simple individuals aggregates into the dynamics of a whole population.

The replicator dynamics takes the limit of a very large, well-mixed population and asks how strategy frequencies change continuously over time. The answer is the replicator equation $\dot{x}_i = x_i(\pi_i - \bar{\pi})$, the continuous-time mathematical form of natural selection under frequency-dependent fitness.

The Moran process takes a small, finite population seriously. In a small population, chance matters: even a fitter strategy can be lost by bad luck before it spreads. The Moran process models one individual being copied at each time step, and the key question is the *fixation probability*, the chance that a single mutant takes over rather than dying out.

The learning and evolutionary dynamics chapter asks what happens when individuals update strategies by imitation, rational introspection, or best-response. These social-learning mechanisms produce the same large-population limit as biological evolution, but differ in important ways in small populations.

Together these chapters show that the replicator equation is the **universal large-population limit** of all plausible evolutionary and social-learning update rules, and that the Nash equilibria of the stage game are its rest points.

9.3 Exercises

Exercise 9.1:

A population of beetles has two colour morphs: cryptic (C, blends with bark) and conspicuous (K, visible to predators).

1. In a forest environment, cryptic beetles have higher fitness than conspicuous beetles. Sketch the qualitative trajectory of the cryptic frequency starting from $x_C = 0.2$. Which morph eventually dominates?
2. Suppose a wildfire removes the bark and the ground becomes pale and open, while predator behaviour stays the same. Sketch the new trajectory, starting again from $x_C = 0.2$. Which morph dominates now?
3. What does the comparison between the two cases illustrate about the relationship between the environment and the direction of selection?

Exercise 9.2:

Return to the stag antler example from Motivating Example: Why Don't Stags Have Bigger Antlers?.

1. Identify which of Darwin's three conditions (variation, heritability, and differential reproduction) are present in the stag antler scenario. Give a one-sentence justification for each condition.
2. Suppose a genetic study reveals that antler-investment level is **not** heritable: each stag grows antlers to a size determined entirely by random developmental noise, regardless of its father's investment level. Which of Darwin's three conditions would fail? Would natural selection still act on antler investment? Explain.
3. The chapter distinguishes genotype, phenotype, and strategy. In the context of antler investment, identify what each of these terms refers to, and explain why natural selection acts on the phenotype but changes the frequency of genotypes.

Exercise 9.3:

A species of snail has a single gene controlling shell colour. At that locus the allele B (brown) is dominant and the allele y (yellow) is recessive, so the genotypes BB and By give brown shells and yy gives a yellow shell.

1. Name the genotype(s) that produce a brown shell and the genotype that produces a yellow shell. Which genotypes are homozygous, and which is heterozygous?
2. Explain, in terms of genotype and phenotype, how two brown-shelled snails can have a yellow-shelled offspring.
3. Suppose the same gene also affects shell thickness, so that the brown allele happens to produce slightly thicker shells. What is this phenomenon called, and why does it mean that selection on shell colour cannot be considered in isolation?

Exercise 9.4:

For each of the following scenarios, state whether fitness is frequency-dependent and briefly justify your answer:

1. A plant that photosynthesises more efficiently than its neighbours has higher fitness regardless of the plant community around it.
2. A fish that mimics the appearance of a toxic species is protected from predators, but only while the toxic species remains common enough for predators to have learned the association.
3. A bacterium secretes a costly enzyme that benefits all bacteria in the colony. Its fitness advantage over non-secretors depends on how many non-secretors are present in the population.
4. A faster cheetah always catches more prey, independently of what other cheetahs are doing.

9.4 Notable Research

The founding paper of evolutionary game theory is (Maynard Smith & Price, 1973), which introduced the Hawk–Dove game and the concept of an evolutionarily stable strategy. It was simultaneously

a contribution to theoretical biology and a demonstration that Nash equilibrium analysis could be applied to non-rational agents. Maynard Smith later developed the theory in full in his book (Maynard Smith, 1982).

The 1973 paper built on George Price's covariance equation (Price, 1970), which gave a clean algebraic statement of selection, and ran in parallel with William Hamilton's earlier game-theoretic treatment of sex ratios and local mate competition (Hamilton, 1967). Taken together, this body of work transformed mid-twentieth-century evolutionary biology from a purely genetic optimisation framework into one in which strategic interaction plays the central role.

The mathematical link between ESS and the dynamic stability of the replicator equation was established by (Taylor & Jonker, 1978). This paper showed that evolutionary stability (a static, game-theoretic concept) and asymptotic stability under selection dynamics (a dynamical systems concept) are, under broad conditions, equivalent, justifying the use of Nash equilibrium as a prediction for biological populations.

The broader programme connecting population genetics to game theory, and showing that many different biological update rules share the same large-population limit, is surveyed in (Nowak, 2006), which provides extensive biological examples alongside the mathematical theory.

9.5 Conclusion

Natural selection is not rational deliberation, but it produces outcomes that look as though it is. The reason is that evolution is a kind of optimisation, but what it optimises is *reproductive success against the current population*, not a fixed objective. Whenever reproductive success depends on what strategies your neighbours use, you have a game. The payoff matrix of that game is the fitness matrix of the biological population.

The three conditions for natural selection (variation, heritability, differential reproduction) translate directly into the components of the mathematical models in the chapters that follow:

- **Variation**: multiple strategies coexist in the population.
- **Heritability**: offspring adopt the parent strategy.
- **Differential reproduction**: strategies that do better than average increase in frequency.

The evolutionary chapters of this book can be read as the mathematical consequences of applying these three conditions to populations playing games.

Table 9.2 summarises the key concepts introduced in this chapter.

Concept	Description	Connects to
Natural selection	Differential reproduction of heritable variants	All evolutionary chapters
Genotype	Inherited genetic specification; determines available strategies	Strategy type in EGT
Phenotype	Observable expression of genotype visible to selection	Strategy played
Allele	Variant form of a gene; distinct alleles correspond to distinct strategies	Strategy in EGT
Fitness W_i	Expected reproductive output of type i	Payoff π_i in evolutionary game theory

Frequency-dependent selection	Fitness depends on population composition	Payoff matrix M_{ij}
Hawk–Dove game	Canonical model of animal conflict	Replicator dynamics, ESS
Evolutionarily stable strategy	Nash equilibrium robust to rare invasion	Stable fixed points of replicator dynamics

Table 9.2: Summary of biological foundations

Attention

The key conceptual move in evolutionary game theory is replacing fixed fitness constants with frequency-dependent payoffs. Once fitness is a payoff that depends on interactions, every result from classical game theory (Nash equilibrium, mixed strategies, evolutionary stability) becomes a statement about which strategies natural selection will sustain.

9.6 Solutions

Solution 9.1: Solution to Exercise 1

1. The cryptic frequency rises monotonically from $x_C = 0.2$ toward fixation, levelling off near $x_C = 1$. With cryptic beetles always reproducing above the population average, their share grows in every generation, slowly at first while they are rare, then more rapidly, then slowing again as they approach fixation.
2. The roles reverse. Against a pale, open background it is now the conspicuous morph that blends in, so it has the higher fitness. Starting from $x_C = 0.2$, the cryptic frequency falls toward zero and the conspicuous morph fixes.
3. There is no universally fitter type. The same colour is advantageous in one environment and deleterious in another; what selection optimises is reproductive success in the *current* environment, not a fixed objective. When the environment changes, the direction of selection can reverse.

Solution 9.2: Solution to Exercise 2

1. All three conditions are present in the stag antler scenario.
   - **Variation.** Stags differ in their antler-investment level: some invest heavily in growth, others minimally.
   - **Heritability.** The tendency to invest in antler growth is partly inherited; offspring resemble their fathers in investment level more than they resemble a randomly chosen stag.
   - **Differential reproduction.** Stags with larger antlers win more contests, gain access to more females, and leave more offspring than stags with smaller antlers.
2. Without heritability the second condition fails. Natural selection requires that fitness differences translate into changes in the composition of the next generation. If antler investment is entirely random and non-heritable, a high-investing stag may leave more offspring but

those offspring do not inherit the high-investment tendency. The trait cannot increase in frequency across generations, so natural selection cannot act on it.

3. The **genotype** is the inherited genetic programme that specifies a stag's antler-investment level: the alleles at the relevant loci that are passed from parent to offspring. The **phenotype** is the actual antler size that results when that programme is expressed, the structure visible to predators and rival stags. The **strategy** in the game-theoretic sense is the inherited behavioural tendency itself (invest heavily vs. invest lightly), which corresponds to the genotype. Natural selection acts on the phenotype because contest outcomes depend on actual antler size, not on the underlying genetics. But it is the genotype that is copied into the next generation, so selection on the phenotype translates into a change in the frequency of genotypes in the population.

Solution 9.3: Solution to Exercise 3

1. The brown phenotype is produced by BB and By; the yellow phenotype only by yy. The genotypes BB and yy are homozygous (two copies of the same allele), and By is heterozygous (one copy of each).
2. A brown-shelled snail can be heterozygous, By: it shows the brown phenotype because B is dominant, yet it still carries the recessive y allele. If two By snails breed, an offspring can inherit y from each parent and so be yy, which is yellow. The phenotype hides the underlying genotype, so the recessive allele persists, unseen, in heterozygotes.
3. This is **pleiotropy**: one gene influencing more than one trait. Because the brown allele affects both colour and thickness, any selection on shell colour automatically shifts the distribution of shell thickness as well. The two traits are tied together through the same allele and cannot respond to selection independently.

Solution 9.4: Solution to Exercise 4

1. **Not frequency-dependent.** The plant's improved photosynthesis gives a fixed advantage regardless of what other plants are doing. Fitness here is a constant property of the genotype, not a function of the population composition.
2. **Frequency-dependent.** The protection conferred by mimicry depends on how common the toxic model species is in the population. When the model is rare, predators have not learned the association, and the mimic gains little benefit. The fitness of the mimicry strategy depends on the frequency of the toxic model in the broader community, rather than on a constant property of the mimic itself.
3. **Frequency-dependent.** The fitness advantage of the secreting bacterium depends on the population composition. When non-secretors are rare, the enzyme cost is borne by nearly everyone and provides little competitive advantage; when non-secretors are common, secretors are exploited but the enzyme still raises mean colony output. The key point is that the fitness of the secreting strategy changes with the frequency of non-secretors in the population, which is precisely what frequency dependence means.
4. **Not frequency-dependent** (under the stated assumption). If faster cheetahs always catch more prey independently of what other cheetahs do, fitness is a constant function of speed and does not depend on the population composition. In practice, cheetah speed might

interact with prey evolution (an arms race), but the scenario as stated implies constant fitness.

10. Replicator Dynamics

When strategies that perform well spread and those that perform poorly shrink, populations evolve. This chapter derives the replicator dynamics equation, which models this selection process mathematically, and analyses the equilibria and long-run behaviour it produces.

10.1 Motivating Example: Coffee clubs and the common good game

Across a large university campus, students form informal coffee clubs. Each day, students join clubs to share a cafetière of coffee and discuss algebraic topology or the latest student gossip. There are three types of students:

- **Cooperators** always bring a scoop of ground coffee to the club.
- **Defectors** show up empty-handed but still drink the coffee.
- **Loners** prefer to skip clubs and enjoy a quiet cup alone, reliable, if a bit less exciting.

In each club, all coffee brought is pooled, brewed, and the resulting pot is shared equally among the attendees. If there are k cooperators, the pot yields rk units of value to be divided among all participants (cooperators and defectors). The loners, who don't attend clubs, receive a fixed payoff σ.

Let the total population be normalized to 1, with:

- x_c the proportion of cooperators,
- x_d the proportion of defectors,
- $x_z = 1 - x_c - x_d$ the proportion of loners.

Assuming the population is large and well-mixed, we can model the average payoff to each strategy as:

- $f_c(x) = r \cdot \frac{x_c}{x_c + x_d} - 1$
- $f_d(x) = r \cdot \frac{x_c}{x_c + x_d}$
- $f_l(x) = \sigma$

Here, $r > 1$ is the return factor of the public good (coffee), and the subtraction of 1 from f_C reflects the cost of bringing coffee.

These payoffs are used in the **replicator dynamics equation**, which models how the frequency of each strategy changes over time:

$$\dot{x}_i = x_i(f_i(x) - \phi) \tag{10.1}$$

where ϕ is the population average payoff.

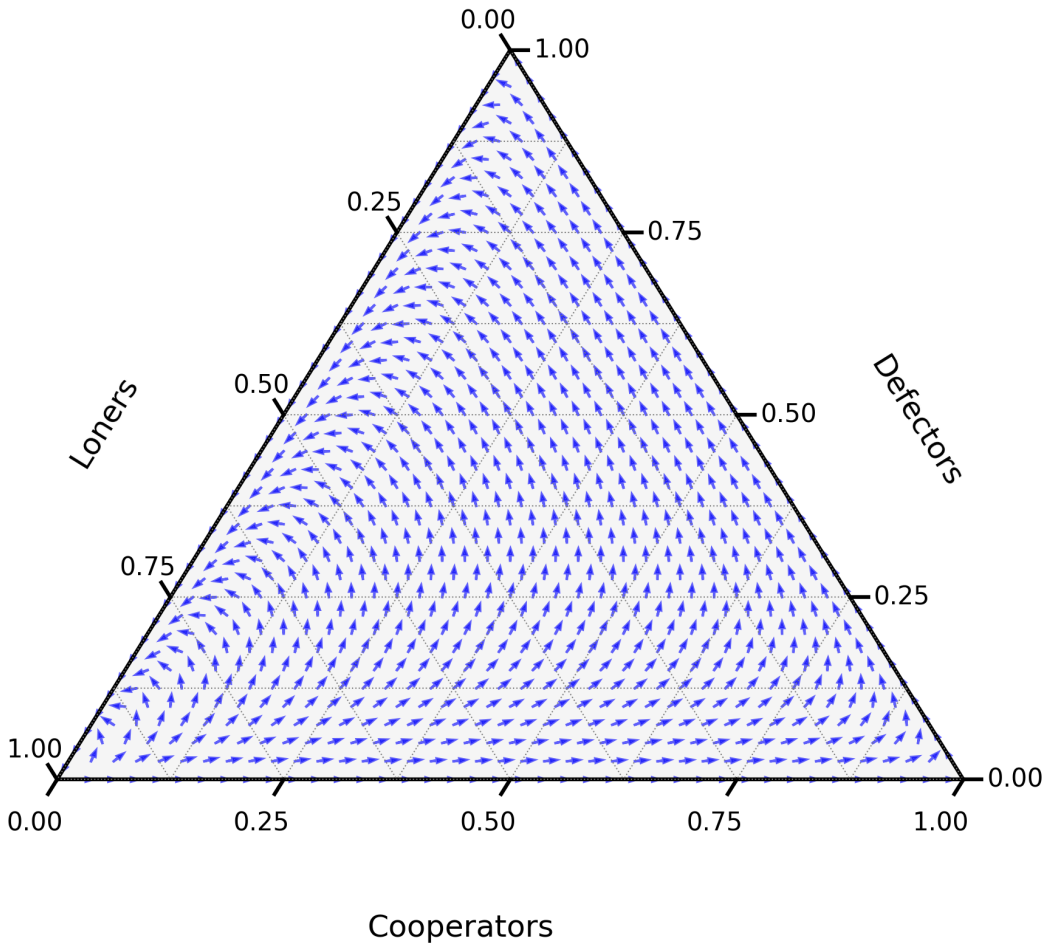


Figure 10.1: A diagram showing the direction of the derivative as given by (10.1) for $r = 3$ and $\sigma = .6$.

Strategies that do better than the average will increase in frequency; those that do worse will decline. This feedback mechanism drives the evolution of behaviour in the population, capturing the shifting fortunes of cooperators, defectors, and loners on campus.

As we will see, this system often exhibits **cyclical behaviour**: when most students cooperate, defectors thrive; when defection becomes too common, students prefer to be loners; when clubs are small and rare, cooperation becomes appealing again. These cycles, and their stability, are precisely what replicator dynamics helps us understand.

Note

The motivating examples in this chapter use human social populations (coffee clubs, student behaviour). The same mathematics applies directly to biological populations; strategies spread through differential reproduction rather than imitation. For the biological interpretation and the connection to natural selection, see The Biological Origins of Evolutionary Game Theory.

10.2 Theory

10.2.1 Definition: Replicator Dynamics Equation

Consider a population with N types of individuals. Let x_i denote the proportion of individuals of type i, so that $\sum_{i=1}^{N} x_i = 1$.

Suppose the fitness of an individual of type i depends on the state of the entire population, through a population-dependent fitness function $f_i(x)$.

The replicator dynamics equation is given by:

$$\frac{dx_i}{dt} = x_i(f_i(x) - \phi) \quad \text{for all } i \tag{10.2}$$

where the average population fitness ϕ is defined as:

$$\phi = \sum_{i=1}^{N} x_i f_i(x) \tag{10.3}$$

This equation describes the change in frequency of each type as proportional to how much better (or worse) its fitness is compared to the population average.

10.2.1.1 Example: The common good game

In the common good game around coffee clubs the replicator dynamics equation can be written as:

$$\begin{aligned} \dot{x}_c &= x_c\left(\frac{rx_c}{x_c + x_d} - 1 - \phi\right) \\ \dot{x}_d &= x_d\left(\frac{rx_c}{x_c + x_d} - \phi\right) \\ \dot{x}_l &= x_l(\sigma - \phi) \end{aligned} \tag{10.4}$$

where:

$$\begin{aligned} \phi &= x_c\left(\frac{rx_c}{x_c + x_d} - 1\right) + x_d\left(\frac{rx_c}{x_c + x_d}\right) + x_l\sigma \\ &= \frac{rx_c^2}{x_c + x_d} - x_c + \frac{rx_cx_d}{x_c + x_d} + x_l\sigma \\ &= \frac{rx_c^2 - x_c(x_c + x_d) + rx_cx_d + x_l\sigma(x_c + x_d)}{x_c + x_d} \\ &= \frac{rx_c(x_c + x_d) - x_c(x_c + x_d) + x_l\sigma(x_c + x_d)}{x_c + x_d} \\ &= \frac{(x_c + x_d)(rx_c - x_c + x_l\sigma)}{x_c + x_d} \\ &= x_c(r - 1) + x_l\sigma \end{aligned} \tag{10.5}$$

10.2.2 Definition: Stable Population

For a given population game with N types of individuals and fitness functions f_i a stable population $\tilde{x}$ is one for which $\dot{x}_i = 0$ for all i.

For a stable population $\tilde{x}_i$:

$$\dot{x}_i = \tilde{x}_i(f_i(\tilde{x}_i) - \phi) = 0 \tag{10.6}$$

so either: $\tilde{x}_i = 0$ or $f_i(\tilde{x}_i) = \phi$.

10.2.2.1 Example: No interior point stable population in the common goods game

For the common good game around coffee clubs we have some immediate stable populations:

- $x = (x_c, x_d, x_l) = (0, 0, 1)$: indeed $\dot{x}_c = \dot{x}_d = 0$ so that $\phi = \sigma$ and so $\dot{x}_l = 0$.
- $x = (x_c, x_d, x_l) = (0, 1, 0)$: indeed $\dot{x}_c = \dot{x}_l = 0$ so that $\phi = f_d(x) = 0$ and so $\dot{x}_d = 0$.
- $x = (x_c, x_d, x_l) = (1, 0, 0)$: indeed $\dot{x}_d = \dot{x}_l = 0$ so that $\phi = f_c(x) = r - 1$ and so $\dot{x}_c = 0$.

A question remains, is there a point in the interior of the simplex of Figure 10.1 that is stable? Such a point has $x_c > 0$, $x_d > 0$ and $x_l > 0$ which implies:

$$f_c(x) = f_d(x) = f_l(x) \tag{10.7}$$

This is not possible as: $f_c(x) = f_d(x) - 1$ for all x.

10.2.3 Definition: Fitness of a strategy in a population

In a population with N types let $f(y, x)$ denote the fitness of an individual playing strategy y in a population x:

$$f(y, x) = \sum_{i=1}^{N} y_i f_i(x) \tag{10.8}$$

10.2.3.1 Example: A new student in the Common Goods Game

For the common good game if we consider a stable population $x = (0, 1, 0)$ where everyone is defecting and assume that a new student enters planning to cooperate 50% of the time and defect 50% of the time, their fitness is given by:

$$f((1/2, 1/2, 0), (0, 1, 0)) = 1/2 f_c(0, 1, 0) + 1/2 f_d(0, 1, 0) = 1/2 \cdot (-1) + 1/2 \cdot 0 = -1/2 \tag{10.9}$$

10.2.4 Definition: Post Entry Population

For a population with N types of individuals Given a population $x \in \mathbb{R}^N_{[0,1]}$ (with $\sum_{i=1}^{N} x_i = 1$), some $\varepsilon > 0$ and a strategy $y \in \mathbb{R}^N_{[0,1]}$ (with $\sum_{i=1}^{N} y_i = 1$), the post entry population x_ε is given by:

$$x_\varepsilon = x + \varepsilon(y - x) \tag{10.10}$$

10.2.4.1 Example: Post Entry Population for the Common Goods Game

For the common good game if we consider the **stable** population $x = (0, 1, 0)$ where everyone is defecting and assume that a new student enters the population planning to cooperate with a coffee club 50% of the time and defect 50% of the time the post entry population will be:

$$x_\varepsilon = (0, 1, 0) + \varepsilon((1/2, 1/2, 0) - (0, 1, 0)) = (0, 1, 0) + (\varepsilon/2, -\varepsilon/2, 0) = (\varepsilon/2, 1 - \varepsilon/2, 0) \tag{10.11}$$

> **Note**
>
> Note that for an infinitely large population a single student is represented by an infinitesimal amount ε.

What is of interest in the field of evolutionary game theory is what happens to the post entry population: does this new student change the stability of the system or is the system going to go back to all students defecting? This is the question of **evolutionary stability**. The evolutionarily stable strategy (ESS), defined formally below, captures exactly this: a population is evolutionarily stable if residents always outperform any rare mutant. For pairwise interaction games the invasion condition reduces to a convenient algebraic characterisation, also developed below.

10.2.5 Definition: Pairwise Interaction Game

Given a population of N types of individuals and a payoff matrix $M^{n \times n}$ a pairwise interaction game is a game where the fitness is $f_i(x)$ is given by:

$$f_i(x) = \sum_{j=1}^{N} x_j M_{ij} \tag{10.12}$$

This corresponds to a population where all individuals interact with all other individuals in the population and obtain a fitness given by the matrix M.

Note that there is a linear algebraic equivalent to (10.12):

$$f = xM \tag{10.13}$$

and then:

$$\phi = xf \tag{10.14}$$

10.2.5.1 Example: The Hawk Dove Game

Consider a population of animals. These animals, when they interact, will always share their food. Due to a genetic mutation, some of these animals may act in an aggressive manner and not share their food. If two aggressive animals meet, they both compete and end up with no food. If an aggressive animal meets a sharing one, the aggressive one will take most of the food.

These interactions can be represented using the matrix A:

$$M = \begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix} \tag{10.15}$$

In this scenario: what is the likely long-term effect of the genetic mutation?

Over time will:

- The population resist the mutation and all the animals continue to share their food.
- The population get taken over by the mutation and all animals become aggressive.
- A mix of animals are present in the population: some act aggressively and some share.

To answer this question, we will model it as a pairwise interaction game with $x \in \mathbb{R}^2_{[0,1]}$ representing the population distribution. In this case:

- x_1 represents the proportion of the population that shares.
- x_2 represents the proportion of the population that acts aggressively.

In this case, the replicator dynamics equation becomes:

$$\begin{aligned} \frac{dx_1}{dt} &= x_1(2x_1 + x_2 - \phi) \\ \frac{dx_2}{dt} &= x_2(3x_1 - \phi) \end{aligned} \tag{10.16}$$

$$\phi = x_1(2x_1 + x_2) + x_2(3x_1) \tag{10.17}$$

Note that $x_2 = 1 - x_1$ so for simplicity of notation we will only use x to represent the proportion of the population that shares.

Thus we can write the single differential equation:

$$\begin{aligned} \frac{dx}{dt} &= x(2x + (1-x) - \phi) \\ &= x(2x + (1-x) - x(2x + (1-x)) - (1-x)(3x)) \\ &= x(2x^2 - 3x + 1) \\ &= x(x-1)(2x-1) \end{aligned} \tag{10.18}$$

We see that there are 3 stable populations:

- $x = 0$: No sharers

- $x = 1$: Only sharers
- $x = 1/2$: A mix of both sharers and aggressive individuals.

This differential equation can then be solved numerically, for example using Euler's Method to show the evolution of

Let us do this with a step size $h = .1$ and an initial population of $x = 3/5$:

Recall, we use the update rule:

$$x_{n+1} = x_n + h \cdot f(x_n) \tag{10.19}$$

to give Table 10.1.

n	t_n	x_n	$f(x_n)$	x_{n+1}
0	0.0	0.600	−0.048	0.595
1	0.1	0.595	−0.046	0.591
2	0.2	0.591	−0.044	0.586
3	0.3	0.586	−0.042	0.582
4	0.4	0.582	−0.040	0.578
5	0.5	0.578	−0.038	0.574
6	0.6	0.574	−0.036	0.571
7	0.7	0.571	−0.035	0.567
8	0.8	0.567	−0.033	0.564
9	0.9	0.564	−0.031	0.561
10	1.0	0.561	−0.030	0.558

Table 10.1: Step by step application of Euler's method to the Hawk Dove game with step size $h = .1$ and $x_0 = 3/5$.

If we repeat this with $x = 2/5$ we obtain Table 10.2.

n	t_n	x_n	$f(x_n)$	x_{n+1}
0	0.0	0.400	0.048	0.405
1	0.1	0.405	0.046	0.409
2	0.2	0.409	0.044	0.414
3	0.3	0.414	0.042	0.418
4	0.4	0.418	0.040	0.422
5	0.5	0.422	0.038	0.426
6	0.6	0.426	0.036	0.429
7	0.7	0.429	0.035	0.433
8	0.8	0.433	0.033	0.436
9	0.9	0.436	0.031	0.439
10	1.0	0.439	0.030	0.442

Table 10.2: Step by step application of Euler's method to the Hawk Dove game with step size $h = .1$ and $x_0 = 2/5$.

It looks the population is converging to the population which has a mix of both sharers and aggressive types: $x = 1/2$. Figure 10.2 confirms this.

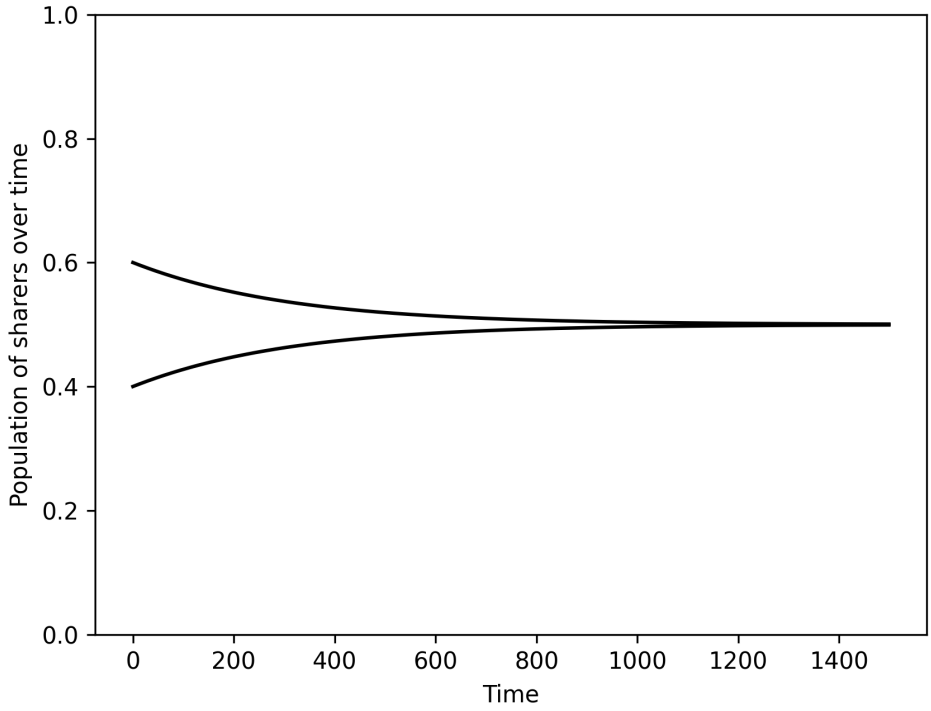


Figure 10.2: The numerical integration of the differential equation (10.18) with two different initial values of x.

This indicates that $x = 1/2$ is an evolutionary stable strategy. To confirm this we could apply the definition of an evolutionarily stable strategy directly; however, for pairwise interaction games there is a theoretic result that can be used instead. We first state the definition formally.

10.2.6 Definition: Evolutionarily Stable Strategy

A strategy σ^* is an **evolutionarily stable strategy (ESS)** if there exists $\bar{\varepsilon} > 0$ such that for all $\sigma' \neq \sigma^*$ and all $0 < \varepsilon < \bar{\varepsilon}$,

$$\big(f\!\!\!\sigma^*, (1-\varepsilon)\sigma^* + \varepsilon\sigma'\big) > \big(f\!\!\!\sigma', (1-\varepsilon)\sigma^* + \varepsilon\sigma'\big) \tag{10.20}$$

where $(1-\varepsilon)\sigma^* + \varepsilon\sigma'$ is the post-entry population: a population consisting mostly of residents with a small fraction ε of mutants.

The condition says that the resident strategy does strictly better than any rare mutant, so selection removes the mutant and the population returns to σ^*. Letting $\varepsilon \to 0$ gives $f(\sigma^*, \sigma^*) \geq f(\sigma', \sigma^*)$ for all σ', the Nash condition for symmetric games. Every ESS is therefore a symmetric Nash equilibrium, but not every Nash equilibrium is an ESS: ESS is a refinement that rules out equilibria vulnerable to invasion.

10.2.7 Theorem: Characterisation of ESS in two-player games

Let σ^* be a strategy in a symmetric two-player game (so that $M_r = M_c^\top$). Then σ^* is an evolutionarily stable strategy (ESS) if and only if, for all $\sigma \neq \sigma^*$, one of the following conditions holds:

1. $f(\sigma^*, \sigma^*) > f(\sigma, \sigma^*)$
2. $f(\sigma^*, \sigma^*) = f(\sigma, \sigma^*)$ and

$f(\sigma^*, \sigma) > f(\sigma, \sigma)$

Conversely, if either of the above conditions holds for all $\sigma \neq \sigma^*$, then σ^* is an ESS in the corresponding population game.

10.2.7.1 Proof

Assume σ^* is an ESS. Then, by definition, there exists $\varepsilon > 0$ such that for all $\sigma \neq \sigma^*$ and all $0 < \varepsilon < \varepsilon$, we have:

$$f(\sigma^*, \chi_\varepsilon) > f(\sigma, \chi_\varepsilon) \tag{10.21}$$

where $\chi_\varepsilon = (1 - \varepsilon)\sigma^* + \varepsilon\sigma$ is the mixed population state. Substituting into the expected fitness, we obtain:

$$(1 - \varepsilon)f(\sigma^*, \sigma^*) + \varepsilon f(\sigma^*, \sigma) > (1 - \varepsilon)f(\sigma, \sigma^*) + \varepsilon f(\sigma, \sigma) \tag{10.22}$$

Rearranging, this inequality holds for all sufficiently small $\varepsilon > 0$ if either:

- $f(\sigma^*, \sigma^*) > f(\sigma, \sigma^*)$ (so the left-hand side is already greater); or
- $f(\sigma^*, \sigma^*) = f(\sigma, \sigma^*)$ but $f(\sigma^*, \sigma) > f(\sigma, \sigma)$, so the second-order term dominates as $\varepsilon \to 0$.

For the converse, suppose neither condition holds. Then either:

- $f(\sigma^*, \sigma^*) < f(\sigma, \sigma^*)$, so for small ε the inequality fails and σ^* is not stable, or
- $f(\sigma^*, \sigma^*) = f(\sigma, \sigma^*)$ and $f(\sigma^*, \sigma) \leq f(\sigma, \sigma)$, in which case the right-hand side is at least as large as the left for small ε, again contradicting stability.

Hence, the two conditions are necessary and sufficient for evolutionary stability.

This theorem gives us a practical method for identifying ESS:

1. Construct the associated symmetric two-player game.
2. Identify all symmetric Nash equilibria of the game.
3. For each symmetric Nash equilibrium, test the two conditions above.

Note that the first condition is very close to the condition for a strict Nash equilibrium, while the second adds a refinement that removes certain non-strict symmetric equilibria. This distinction is especially important when considering equilibria in strategies.

10.2.7.2 Example: Evolutionary stability in the Hawk-Dove game

Let us consider the **Hawk-Dove** game. The associated symmetric two-player game can be written in a general form. Let v denote the value of the resource and c the cost of conflict with $v < c$.

Row player's payoff matrix:

$$A = \begin{pmatrix} \frac{v-c}{2} & v \\ 0 & \frac{v}{2} \end{pmatrix} \tag{10.23}$$

Column player's payoff matrix:

$$B = \begin{pmatrix} \frac{v-c}{2} & 0 \\ v & \frac{v}{2} \end{pmatrix} \tag{10.24}$$

To find symmetric Nash equilibria, we apply the support enumeration algorithm:

$$\begin{aligned} f(\sigma^*, H) &= f(\sigma^*, D) \\ q^* &= \frac{v}{c} \end{aligned} \tag{10.25}$$

This gives

$$\sigma^* = \left(\frac{v}{c}, 1 - \frac{v}{c}\right) \tag{10.26}$$

where individuals are aggressive with probability $\frac{v}{c}$ and share otherwise.

To determine whether this strategy is evolutionarily stable, we check the conditions of the characterisation theorem.

Since $f(\sigma^*, \sigma^*) = f(\sigma^*, \text{Aggressive}) = f(\sigma^*, \text{Sharing})$, the **first condition** does *not* hold. We must therefore verify the **second condition**:

$$f(\sigma^*, \sigma) > f(\sigma, \sigma) \tag{10.27}$$

Let $\sigma = (\omega, 1 - \omega)$ be an arbitrary strategy. Then:

$$f(\sigma^*, \sigma) = \frac{v}{c} \cdot \omega \cdot \frac{v - c}{2} + \frac{v}{c} \cdot (1 - \omega) \cdot v + \left(1 - \frac{v}{c}\right) \cdot (1 - \omega) \cdot \frac{v}{2} \tag{10.28}$$

$$f(\sigma, \sigma) = \omega^2 \cdot \frac{v - c}{2} + \omega(1 - \omega) \cdot v + (1 - \omega)^2 \cdot \frac{v}{2} \tag{10.29}$$

Subtracting the two expressions gives:

$$\begin{aligned} &f(\sigma^*, \sigma) - f(\sigma, \sigma) \\ &\qquad = \frac{c}{2}\left(\frac{v}{c} - \omega\right)^2 > 0 \end{aligned} \tag{10.30}$$

This inequality is strictly satisfied for all $\omega \neq \frac{v}{c}$, so the second condition holds and σ^* is an evolutionarily stable strategy.

10.2.8 Definition: Replicator Mutator Dynamics Equation

An extension of the replicator equation is to allow for mutation. In this setting, reproduction is imperfect: individuals of a given type can give rise to individuals of another type.

This process is represented by a matrix Q, where Q_{ij} denotes the probability that an individual of type j is produced by an individual of type i.

In this case, the replicator dynamics equation can be modified to yield the **replicator-mutation equation**:

$$\frac{dx_i}{dt} = \sum_{j=1}^{N} x_j f_j Q_{ji} - x_i \phi \quad \text{for all } i \tag{10.31}$$

10.2.8.1 Example: The Replicator Mutator Dynamics Equation for the Hawk Dove Game

Let there be a 10% chance that aggressive individuals will produce sharing ones in which case the matrix Q is given by:

$$Q = \begin{pmatrix} 1 & 0 \\ 1/10 & 9/10 \end{pmatrix} \tag{10.32}$$

10.2.8.2 Example: Recovering the Replicator Dynamics Equation from the Replicator Mutator Dynamics Equation

Show that when $Q = I_N$ (the identity matrix of size N) then the replicator dynamics equation corresponds to the replicator dynamics equation.

The replicator-mutation equation is:

$$\frac{dx_i}{dt} = \sum_{j=1}^{N} x_j f_j Q_{ji} - x_i \phi \quad \text{for all } i \tag{10.33}$$

As $Q = I_N$:

$$Q_{ij} = \tag{10.34}$$

This gives:

$$\begin{aligned} \frac{dx_i}{dt} &= x_i f_i Q_{ii} - x_i \phi \quad \text{for all } i Q_{ij} = 0 \quad \text{for all } i \neq j \\ &= x_i f_i - x_i \phi \quad \text{for all } i \quad Q_{ii} = 1 \\ &= x_i (f_i - \phi) \quad \text{for all } i \end{aligned} \tag{10.35}$$

10.2.9 Definition: Asymmetric Replicator Dynamics Equation

A further extension of the replicator dynamics framework accounts for populations divided into two distinct subsets. Individuals in the first population are one of M possible types, while those in the second population are one of N possible types.

This setting arises naturally in **asymmetric games**, where the roles of the players differ and the strategy sets need not be the same (i.e., $M \neq N$). In such cases, the standard replicator equation does not apply directly.

The **asymmetric replicator dynamics equations** describe the evolution of strategy distributions x and y in each population:

$$\frac{dx_i}{dt} = x_i \Big((f_x)_i - \phi_x \Big) \quad \text{for all } 1 \leq i \leq M \tag{10.36}$$

$$\frac{dy_j}{dt} = y_j \Big((f_y)_j - \phi_y \Big) \quad \text{for all } 1 \leq j \leq N \tag{10.37}$$

Here:

- $(f_x)_i$ and $(f_y)_j$ are the expected fitnesses of type i and j in each population respectively,
- ϕ_x and ϕ_y denote the average fitnesses in the respective populations.

10.2.9.1 Example: Tennis Serve and Return

In tennis, serving and receiving form an asymmetric interaction. The **server** (row player) chooses one of two serves, while the **receiver** (column player) chooses one of three possible return strategies.

> The server can deliver a **power** or **spin** serve. The receiver can either prepare for power, cover a wide spin, or take an early aggressive position.

This leads to an asymmetric game where the server has 2 strategies and the receiver has 3. The game matrices are:

$$M_r = \begin{pmatrix} 3 & 1 & 2 \\ 4 & 2 & 1 \end{pmatrix} \tag{10.38}$$

$$M_c = \begin{pmatrix} 1 & 3 & 2 \\ 0 & 2 & 4 \end{pmatrix} \tag{10.39}$$

These matrices are based on the following assumptions:

- If the server uses a power serve (r_1) and the receiver prepares for it (c_1), the server has some success (payoff 3), but the receiver also does reasonably well (payoff 1).
- If the server tries spin (r_2) and the receiver is covering for it (c_2), the payoff is more balanced (2 for each).
- A mismatch, such as a power serve into a receiver expecting spin (r_1 vs c_2), favors the server more (payoff 1 vs 3).
- Conversely, if the receiver takes an early position and guesses right against spin (r_2 vs c_3), they gain a big advantage (payoff 1 vs 4).

Let $x = (x_1, x_2)$ be the strategy distribution of the server and $y = (y_1, y_2, y_3)$ that of the receiver. The **asymmetric replicator dynamics** for this game are:

$$\frac{dx_i}{dt} = x_i((Ay)_i - x^\top Ay) \quad \text{for } 1 \leq i \leq 2 \tag{10.40}$$

$$\frac{dy_j}{dt} = y_j\Big((x^\top B)_j - x^\top By\Big) \quad \text{for } 1 \leq j \leq 3 \tag{10.41}$$

Figure 10.3 shows the numerical solutions of these differential equations over time.

- Preparing for Power quickly dies out as a strategy;
- There is a cycle with the server changing between power and spin while the returner cycles between preparing for spin and taking an aggressive position.

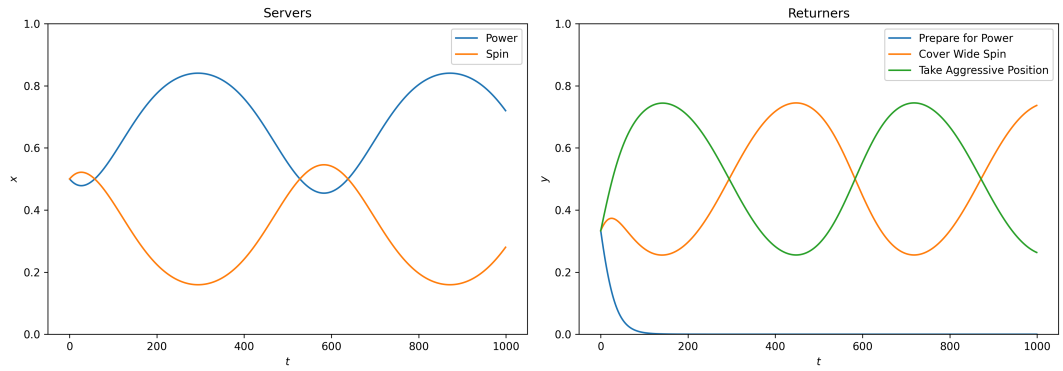


Figure 10.3: Numerical solutions to the asymmetric replicator dynamics equation. Preparing for power quickly dies out as a played strategy in the population. There is a cycle of the 2 remaining strategies for the returner and for the server although power remains the strategy player most often.

10.3 Exercises

Exercise 10.1:

Consider a population with two types of individuals: $x = (x_1, x_2)$ such that $x_1 + x_2 = 1$. Obtain all the stable populations for the system defined by the following fitness functions:

1. $f_1(x) = x_1 - x_2 \qquad f_2(x) = x_2 - 2x_1$

2. $f_1(x) = x_1 x_2 - x_2 \qquad f_2(x) = x_2 - x_1 + \frac{1}{2}$
3. $f_1(x) = x_1^2 \qquad f_2(x) = x_2^2$

For each stable population, choose a nearby post-entry state and solve the replicator dynamics equation numerically to evaluate the evolutionary stability.

Exercise 10.2:

For the following games, obtain all the stable populations for the associated pairwise interaction game:

1. $A = \begin{pmatrix} 2 & 4 \\ 5 & 3 \end{pmatrix}$
2. $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Exercise 10.3:

Consider the pairwise contest games defined by the following associated two-player games. In each case, identify all **evolutionarily stable strategies (ESS)**.

1.

$$M_r = \begin{pmatrix} 2 & 4 \\ 5 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 2 & 5 \\ 4 & 1 \end{pmatrix} \tag{10.42}$$

1.

$$M_r = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad M_c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{10.43}$$

Exercise 10.4:

In a mathematics department, researchers can choose to use one of two systems for type-setting their research papers: LaTeX or Word. We will refer to these two strategies as L and W respectively. A user of W receives a basic utility of 1. As L is more widely used by mathematicians outside the department and is generally considered the superior system, a user of L receives a basic utility of $\alpha > 1$. Since collaboration is common, it is beneficial for researchers to use the same system. If μ denotes the proportion of L users, we define:

$$u(L, \chi) = \alpha + 2\mu \tag{10.44}$$

$$u(W, \chi) = 1 + 2(1 - \mu) \tag{10.45}$$

What are the evolutionarily stable strategies?

Exercise 10.5:

Throughout, the following game is considered:

Users of an online messaging platform can choose one of two incompatible apps: app A or app B. The action set in this game is $\mathcal{A} = \{A, B\}$.

Each user gains a base level of utility from being able to send messages at all. On top of this, their utility increases with the proportion of other users choosing the same app (since it becomes easier to contact friends).

Considering a population vector $x = (x_1, 1 - x_1)$ where x_1 is the proportion of the population using app A, the fitness functions are:

$$f_A(x) = 1 + 2x_1, \qquad f_B(x) = 1 + 2(1 - x_1). \tag{10.46}$$

1. **Give an interpretation for the fitness functions.**
2. **State the replicator dynamics equation for a population with two types.**
3. **Define a stable population for the replicator dynamics.**
4. **Define a post–entry population.**
5. **State the definition of an evolutionarily stable strategy (ESS).**
6. **Give the replicator dynamics equation for this game and obtain all stable populations.**
7. For each of the following initial populations of $x \in \{(0.01, 0.99), (0.49, 0.51), (0.99, 0.01)\}$, calculate the first step of Euler's method of numerical integration with a step size of $h = 0.01$.

In each case offer an interpretation in terms of evolutionary stability.

1. Now suppose that the strength of the coordination benefit is a parameter $a \neq 0$, so that

$$f_A(x) = 1 + ax_1, \qquad f_B(x) = 1 + a(1 - x_1). \tag{10.47}$$

Derive the replicator dynamics for general a and determine for which values of a the mixed population $x^\star = \left(\frac{1}{2}, \frac{1}{2}\right)$ is evolutionarily stable.
Justify your answer.

10.4 Programming

In Appendix: Numerical Integration, we introduce general programming approaches for numerically solving differential equations. These apply directly to the replicator dynamics equation. Here, we focus on tools specifically tailored to pairwise interaction games.

10.4.1 Solving symmetric replicator dynamics

The `Nashpy` library provides built-in functionality for solving the replicator dynamics equation in a pairwise interaction game.

Let us consider the classic **Rock–Paper–Scissors** game:

```
import nashpy as nash
import numpy as np

M_r = np.array([[0, 1, -1], [-1, 0, 1], [1, -1, 0]])
game = nash.Game(M_r)
```

We can compute the population trajectory from an initial distribution:

```
x0 = np.array([1 / 6, 1 / 6, 2 / 3])
timepoints = np.linspace(0, 10, 1500)
```

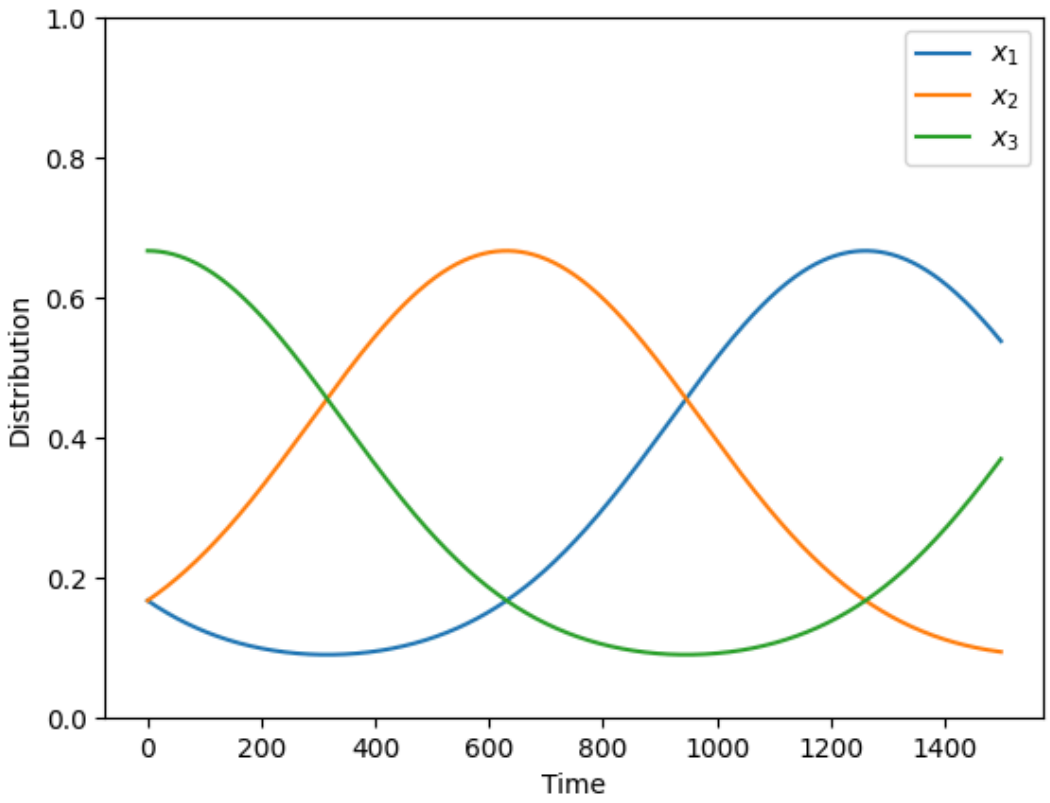
```
xs = game.replicator_dynamics(y0=x0, timepoints=timepoints).T
print(xs)
```

```
[[0.16666667 0.16611198 0.16555975 ... 0.53953201 0.53854678 0.53755904]
 [0.16666667 0.16722383 0.16778347 ... 0.09354312 0.09343612 0.09333053]
 [0.66666667 0.66666419 0.66665678 ... 0.36692487 0.36801711 0.36911043]]
```

To visualize the evolution of strategy frequencies over time:

```
import matplotlib.pyplot as plt

plt.figure()
plt.plot(xs.T)
plt.ylim(0, 1)
plt.legend(["$x_1$", "$x_2$", "$x_3$"])
plt.ylabel("Distribution")
plt.xlabel("Time")
```

```
Text(0.5, 0, 'Time')
```



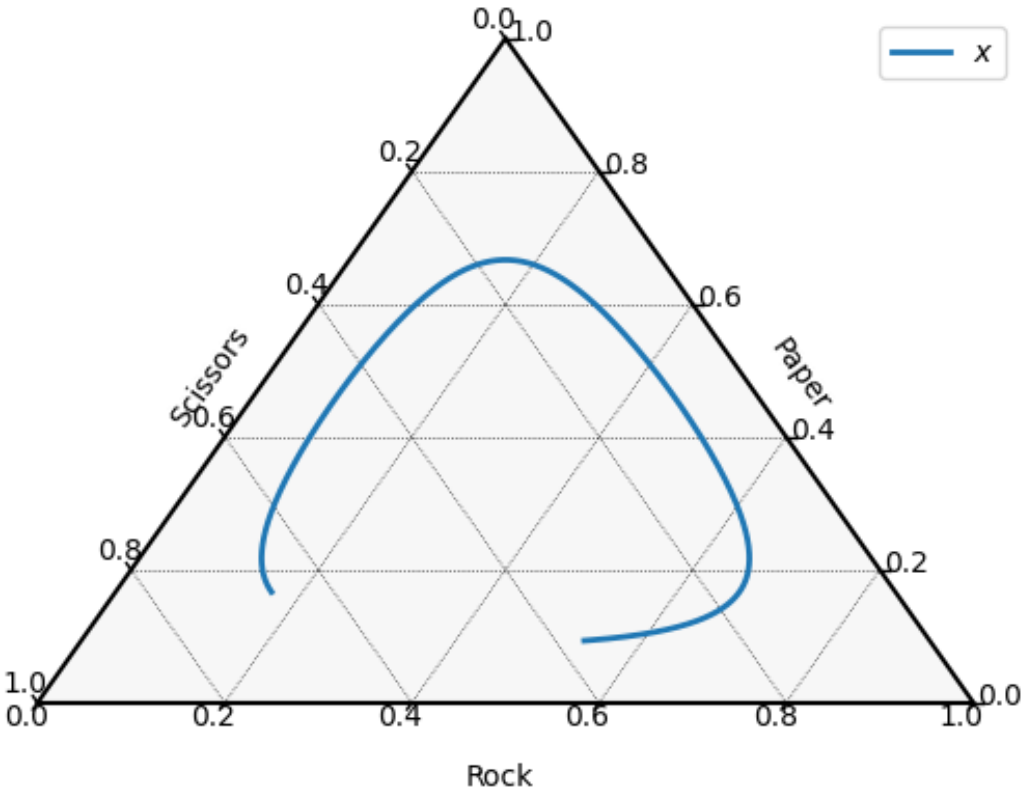
10.4.2 Plotting a simplex with ternary

The ternary library (Harper, 2019) allows for plotting trajectories on a simplex, ideal for representing three-component distributions that sum to one.

We can use it to plot the Rock–Paper–Scissors trajectory:

```
import ternary
```

```
figure, tax = ternary.figure(scale=1.0)
tax.boundary()
tax.gridlines(multiple=0.2, color="black")
# Plot the data
tax.plot(xs.T, linewidth=2.0, label="$x$")
tax.ticks(axis='lbr', multiple=0.2, linewidth=1, tick_formats="%.1f")
tax.legend()
tax.left_axis_label("Scissors")
tax.right_axis_label("Paper")
tax.bottom_axis_label("Rock")
tax.ax.axis('off')
tax.show()
```



10.4.3 Solving Asymmetric Replicator dynamics

The Nashpy library also supports numerical solutions for the asymmetric replicator dynamics equation.

```
M_r = np.array([[3, 1, 2], [4, 2, 1]])
M_c = np.array([[1, 3, 2], [0, 2, 4]])
game = nash.Game(M_r, M_c)

x0 = np.array([1/2, 1/2])
y0 = np.array([1/3, 1/3, 1/3])
timepoints = np.linspace(0, 20, 1000)

xs, ys = game.asymmetric_replicator_dynamics(x0=x0, y0=y0, timepoints=timepoints)
print(xs)
```

```
[[0.5        0.5       ]
 [0.49836507 0.50163493]
 [0.49679694 0.50320306]
 ...
 [0.72372346 0.27627654]
 [0.7218324  0.2781676 ]
 [0.7199282  0.2800718 ]]
```

The corresponding trajectory for the column player's strategy distribution:

```
print(ys)
```

```
[[ 3.33333333e-01  3.33333333e-01  3.33333333e-01]
 [ 3.23391408e-01  3.36602736e-01  3.40005857e-01]
 [ 3.13590591e-01  3.39735869e-01  3.46673540e-01]
 ...
 [-1.06554178e-13  7.35381389e-01  2.64618611e-01]
 [-1.25688203e-13  7.36036679e-01  2.63963321e-01]
 [-1.40730981e-13  7.36668819e-01  2.63331181e-01]]
```

10.5 Notable Research

The original conceptual idea of an **evolutionarily stable strategy (ESS)** was formulated by Maynard Smith (Maynard Smith, 1982; Maynard Smith & Price, 1973). Although these works did not explicitly introduce the replicator dynamics equation, they were foundational in connecting game theory with evolutionary biology.

The first formal presentation of the **replicator dynamics equation** appeared in (Taylor & Jonker, 1978), which directly built upon Maynard Smith's ESS framework. This formulation was later extended to multi-player games in (Palm, 1984), and to **asymmetric populations** in (Accinelli & Carrera, 2011).

Several influential applications of replicator dynamics have since emerged. For example, (Komarova et al., 2001) used replicator-mutator dynamics to model the spread of grammatical structures in language populations. In the context of cooperation, (Hilbe et al., 2013) applied the model to study the evolution of reactive strategies, while (Knight et al., 2024) recently demonstrated how extortionate strategies fail to persist under evolutionary pressure.

A particularly notable extension is found in (Weitz et al., 2016), where the game itself changes dynamically depending on the population state. This approach is especially relevant in modelling the **tragedy of the commons** and other environmental feedback systems.

In (Lv et al., 2023), a model similar to the one in Section: Motivating Example is examined using both replicator dynamics and a **discrete population model**. The latter is explored in detail in Chapter: Moran Process. Remarkably, the replicator dynamics equation emerges as the **infinite-population limit** of the discrete model, a connection rigorously established in (Traulsen et al., 2005).

10.6 Conclusion

The replicator dynamics equation provides a powerful lens through which to study strategy evolution in large populations. By linking the fitness of strategies to their growth or decline in the population, it captures the essence of selection and adaptation.

Throughout this chapter, we explored how replicator dynamics:

- arise naturally in settings like public goods provision (e.g., coffee clubs),

- describe population change in terms of differential equations,
- connect with the concept of **evolutionarily stable strategies (ESS)**,
- extend to incorporate mutation and asymmetry,
- and can be simulated and visualized with numerical methods and tools.

From modelling simple two-strategy contests to rich three-strategy dynamics on a simplex, replicator dynamics offer an interpretable and analytically rich framework for evolutionary game theory. Table Table 10.3 gives a summary of the main concepts of this chapter.

Concept	Description
Replicator Dynamics Equation	Models strategy frequency change based on relative fitness
Average Population Fitness (ϕ)	Weighted average of individual fitnesses
Stable Population	A distribution where no strategy's frequency changes over time
Evolutionarily Stable Strategy (ESS)	A stable strategy resistant to invasion by nearby alternatives
Post Entry Population	Perturbed population after a rare mutant enters
Replicator-Mutator Equation	Extension accounting for imperfect strategy transmission
Asymmetric Replicator Dynamics	Models evolution in multi-population or role-asymmetric settings
Pairwise Interaction Game	Fitness determined by payoffs in repeated pairwise interactions

Table 10.3: Summary of key concepts in replicator dynamics.

Important

A strategy grows when its fitness exceeds the population average, and declines otherwise.

This central insight, encoded in the replicator dynamics equation, allows us to understand not just what equilibria exist, but how populations evolve toward them (or cycle away from them) over time.

10.7 Solutions

Solution 10.1: Solution to Exercise 1

For a population with two types the replicator dynamics reduce to a single equation. Writing $x = x_1$ and $x_2 = 1 - x$,

$$\dot{x} = x \left(f_1(x) - \phi \right), \qquad \phi = x f_1(x) + (1-x) f_2(x), \tag{10.48}$$

so that

$$\dot{x} = x(1-x) \left(f_1(x) - f_2(x) \right). \tag{10.49}$$

A population is stable when $\dot{x} = 0$, that is when $x = 0$, $x = 1$, or $f_1(x) = f_2(x)$. The two monomorphic populations $x = 0$ and $x = 1$ are always stable; any interior stable population solves $f_1(x) = f_2(x)$.

1. Here $f_1 - f_2 = (x_1 - x_2) - (x_2 - 2x_1) = 5x - 2$, which vanishes at $x = 2/5$. The stable populations are $x \in \{0, 2/5, 1\}$. Since $f_1 - f_2 < 0$ for $x < 2/5$ and $f_1 - f_2 > 0$ for $x >$

$2/5$, the flow moves away from $2/5$ towards the nearer endpoint: the interior population is unstable and the two monomorphic populations are evolutionarily stable.

2. Here $f_1 = x_2(x_1 - 1) = -(1-x)^2$ and $f_2 = 3/2 - 2x$, so $f_1 - f_2 = -x^2 + 4x - 5/2$. The only root in $[0,1]$ is $x = 2 - \sqrt{6}/2 \approx 0.775$ (the other root exceeds 1). The stable populations are $x \in \left\{0, 2 - \sqrt{6}/2, 1\right\}$. The interior root is again unstable, since $f_1 - f_2$ increases through zero there and nearby states are pushed to an endpoint.
3. Here $f_1 - f_2 = x_1^2 - x_2^2 = x_1 - x_2 = 2x - 1$, which vanishes at $x = 1/2$. The stable populations are $x \in \{0, 1/2, 1\}$, with the interior mixed population unstable and the two monomorphic populations stable.

In every case the interior stable population is a repellor and only the monomorphic populations survive perturbation. We confirm this by integrating the replicator dynamics from post-entry states either side of each interior population.

```
import numpy as np

def replicator_trajectory(f_1, f_2, initial_x, step_size=0.01, steps=2000):
    trajectory = [initial_x]
    x = initial_x
    for _ in range(steps):
        average_fitness = x * f_1(x) + (1 - x) * f_2(x)
        x = x + step_size * x * (f_1(x) - average_fitness)
        trajectory.append(x)
    return np.array(trajectory)

fitness_pairs = {
    1: (lambda x: x - (1 - x), lambda x: (1 - x) - 2 * x),
    2: (lambda x: x * (1 - x) - (1 - x), lambda x: (1 - x) - x + 1 / 2),
    3: (lambda x: x ** 2, lambda x: (1 - x) ** 2),
}
interior_population = {1: 2 / 5, 2: 2 - np.sqrt(6) / 2, 3: 1 / 2}
for part, (f_1, f_2) in fitness_pairs.items():
    from_below = replicator_trajectory(f_1, f_2, interior_population[part] -
0.02)[-1]
    from_above = replicator_trajectory(f_1, f_2, interior_population[part] +
0.02)[-1]
    print(f"part {part}: from below -> {from_below:.3f}, from above ->
{from_above:.3f}")
```

```
part 1: from below -> 0.000, from above -> 1.000
part 2: from below -> 0.000, from above -> 1.000
part 3: from below -> 0.000, from above -> 1.000
```

Solution 10.2: Solution to Exercise 2

For a pairwise interaction game with two types the fitness functions are $f_1(x) = A_{11}x_1 + A_{12}x_2$ and $f_2(x) = A_{21}x_1 + A_{22}x_2$. As in the previous solution the dynamics reduce to $\dot{x} = x(1-x)(f_1 - f_2)$ with $x = x_1$ and

$$f_1 - f_2 = (A_{11} - A_{21})x + (A_{12} - A_{22})(1 - x). \tag{10.50}$$

1. For $A = \left(\begin{smallmatrix} 2 & 4 \\ 5 & 3 \end{smallmatrix}\right)$ we obtain $f_1 - f_2 = -3x + (1-x) = 1 - 4x$, which vanishes at $x = 1/4$. The stable populations are $x \in \{0, 1/4, 1\}$. Here $f_1 - f_2 > 0$ for $x < 1/4$ and $f_1 - f_2 < 0$ for $x > 1/4$, so the flow moves towards the interior population: $x = 1/4$ is stable and the two monomorphic populations are unstable. This is an anti-coordination (Hawk-Dove) game.
2. For $A = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ we obtain $f_1 - f_2 = x - (1-x) = 2x - 1$, which vanishes at $x = 1/2$. The stable populations are $x \in \{0, 1/2, 1\}$. This is a coordination game: the interior population is unstable and the two monomorphic populations are stable.

Solution 10.3: Solution to Exercise 3

We apply the characterisation of ESS for symmetric two-player games. In each case we identify the symmetric Nash equilibria and test the two conditions.

1. Here $A = \left(\begin{smallmatrix} 2 & 4 \\ 5 & 1 \end{smallmatrix}\right)$. Neither pure strategy is a symmetric equilibrium: against residents playing the first strategy a mutant playing the second earns $5 > 2$, and against residents playing the second a mutant playing the first earns $4 > 1$. The only symmetric equilibrium is the fully mixed $\sigma^* = (1/2, 1/2)$, obtained from $f_1 - f_2 = (2-5)x + (4-1)(1-x) = 3 - 6x = 0$. Because σ^* is fully mixed, $f(\sigma^*, \sigma^*) = f(\sigma, \sigma^*)$ for every σ, so the first condition never holds and we test the second. A direct calculation gives $f(\sigma^*, \sigma) - f(\sigma, \sigma) = \frac{3}{2}(2x-1)^2 > 0$ for all $\sigma \neq \sigma^*$, so $\sigma^* = (1/2, 1/2)$ is the unique ESS. This is an anti-coordination game and σ^* is the attractor of the replicator dynamics.
2. Here $A = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ is a coordination game. Both pure strategies are strict symmetric equilibria: against residents playing the first strategy a mutant playing the second earns $0 < 1$, so the first condition holds strictly and $e_1 = (1, 0)$ is an ESS; by symmetry $e_2 = (0, 1)$ is an ESS. The mixed equilibrium $\sigma^* = (1/2, 1/2)$, from $2x - 1 = 0$, is not an ESS: here $f(\sigma^*, \sigma) - f(\sigma, \sigma) = -\frac{1}{2}(2x-1)^2 < 0$, so a mutant does strictly better against the perturbed population and invades. The evolutionarily stable strategies are the two pure conventions.

Solution 10.4: Solution to Exercise 4

Write x for the proportion μ of LaTeX users, so the two fitness functions are $f_L(x) = \alpha + 2x$ and $f_W(x) = 1 + 2(1-x)$. The replicator dynamics for two types reduce to

$$\dot{x} = x(1-x)\,\big(\, f_L(x) - f_W(x) \,\big) = x(1-x)\,\big(\, \alpha - 3 + 4x \,\big). \tag{10.51}$$

The fixed points are $x = 0$ (everyone uses Word), $x = 1$ (everyone uses LaTeX), and the interior point $x^* = (3-\alpha)/4$, which lies in $(0, 1)$ precisely when $1 < \alpha < 3$.

The coefficient of x inside the bracket is positive, so $f_L - f_W$ is increasing: this is a coordination game and any interior fixed point is a repellor. Checking the endpoints:

- all-LaTeX ($x = 1$): a lone Word user among LaTeX users earns $f_W(1) = 1$ against $f_L(1) = \alpha + 2 > 1$, so $x = 1$ is a strict equilibrium and an ESS for every $\alpha > 1$;
- all-Word ($x = 0$): a lone LaTeX user among Word users earns $f_L(0) = \alpha$ against $f_W(0) = 3$, so $x = 0$ is a strict equilibrium, and hence an ESS, if and only if $\alpha < 3$.

Therefore, if $1 < \alpha < 3$ both conventions are evolutionarily stable and the department locks in to whichever system already predominates; the interior mixture $x^* = (3-\alpha)/4$ is unstable and separates the two basins of attraction. If $\alpha \geq 3$ the intrinsic advantage of LaTeX is large enough that all-Word ceases to be stable and LaTeX is the only evolutionarily stable convention.

Solution 10.5: Solution to Exercise 5

1. The fitness of an app user is made up of:

- a **base utility** of using any messaging app at all;
- an **extra utility** that increases with the proportion of users choosing the same app.

For app A:

$$f_A(x) = \underbrace{1}_{\text{base utility of using any app}} + \underbrace{2x_1}_{\text{extra utility from many friends using app } A} \tag{10.52}$$

For app B:

$$f_B(x) = \underbrace{1}_{\text{base utility of using any app}} + \underbrace{2(1-x_1)}_{\text{extra utility from many friends using app } B} \tag{10.53}$$

So users gain more by coordinating on the same app as their friends.

2. If x_1 is the proportion of the population using strategy 1 (here, app A), the replicator dynamics for a two type population is:

$$\frac{dx_1}{dt} = x_1 \left(f_1(x) - \phi(x) \right), \tag{10.54}$$

where the average fitness $\phi(x)$ is

$$\phi(x) = x_1 f_1(x) + (1-x_1) f_2(x). \tag{10.55}$$

3. From Definition: Stable Population:

For a given population game with N types of individuals and fitness functions f_i a stable population $\tilde{x}$ is one for which $\dot{x}_i = 0$ for all i.

4. From Definition: Post Entry Population: For a population with N types of individuals Given a population $x \in \mathbb{R}^N_{[0,1]}$ (with $\sum_{i=1}^N x_i = 1$), some $\varepsilon > 0$ and a strategy $y \in \mathbb{R}^N_{[0,1]}$ (with $\sum_{i=1}^N y_i = 1$), the post entry population x_ε is given by:

$$x_\varepsilon = x + \varepsilon(y - x) \tag{10.56}$$

5. From Definition: Evolutionarily Stable Strategy:

A strategy σ^* is an evolutionarily stable strategy if there exists $\bar{\varepsilon} > 0$ such that for all $\sigma' \neq \sigma^*$ and all $0 < \varepsilon < \bar{\varepsilon}$,

$$\left(f\sigma^*, (1-\varepsilon)\sigma^* + \varepsilon\sigma' \right) > \left(f\sigma', (1-\varepsilon)\sigma^* + \varepsilon\sigma' \right) \tag{10.57}$$

where $(1-\varepsilon)\sigma^* + \varepsilon\sigma'$ is the post-entry population.

6. We have

$$f_A(x) = 1 + 2x_1, \qquad f_B(x) = 1 + 2(1 - x_1). \tag{10.58}$$

The average fitness is

$$\begin{aligned} \phi(x) &= x_1 f_A(x) + (1 - x_1) f_B(x) \\ &= x_1(1 + 2x_1) + (1 - x_1)\ \big(\ 1 + 2(1 - x_1)\ \big) \\ &= x_1 + 2x_1^2 + (1 - x_1)(3 - 2x_1) \\ &= x_1 + 2x_1^2 + 3 - 5x_1 + 2x_1^2 \\ &= 3 - 4x_1 + 4x_1^2. \end{aligned} \tag{10.59}$$

The replicator dynamics is

$$\begin{aligned} \frac{dx_1}{dt} &= x_1\ \big(\ f_A(x) - \phi(x)\ \big) \\ &= x_1\ \big(\ (1 + 2x_1) - (3 - 4x_1 + 4x_1^2)\ \big) \\ &= x_1(-2 + 6x_1 - 4x_1^2) \\ &= -2x_1(2x_1^2 - 3x_1 + 1) \\ &= -2x_1(2x_1 - 1)(x_1 - 1). \end{aligned} \tag{10.60}$$

The fixed points are the solutions of $\frac{dx_1}{dt} = 0$:

$$x_1 \in \left\{0, \frac{1}{2}, 1\right\}. \tag{10.61}$$

The above answers the question, below is some code to confirm:

```
import sympy as sym

x_1 = sym.Symbol("x_1")

f_A = 1 + 2 * x_1
f_B = 1 + 2 * (1 - x_1)
phi = x_1 * f_A + (1 - x_1) * f_B
sym.simplify(phi)
```

```
4*x_1**2 - 4*x_1 + 3
```

```
x_1_dash = x_1 * (f_A - phi)
sym.simplify(x_1_dash)
```

```
2*x_1*(-2*x_1**2 + 3*x_1 - 1)
```

```
x_1_dash = x_1 * (f_A - phi)
sym.solveset(x_1_dash, x_1)
```

```
{0, 1/2, 1}
```

7. We use the replicator equation from Question 6:

$$\begin{aligned}\dot{x}_1 &= \frac{dx_1}{dt} \\ &= -2x_1(2x_1 - 1)(x_1 - 1).\end{aligned} \tag{10.62}$$

Euler's method with step size $h = 0.01$ gives

$$x_1^{(1)} = x_1^{(0)} + h\,\dot{x}_1\big|_{x_1^{(0)}}. \tag{10.63}$$

- (a) Initial population $x = (0.01, 0.99)$

Here $x_1^{(0)} = 0.01$:

$$\begin{aligned}\dot{x}_1\big|_{0.01} &= -2 \cdot 0.01 \cdot (2 \cdot 0.01 - 1) \cdot (0.01 - 1) \\ &= -2 \cdot 0.01 \cdot (-0.98) \cdot (-0.99) \\ &\approx -0.019404.\end{aligned} \tag{10.64}$$

Thus

$$\begin{aligned}x_1^{(1)} \\ = 0.01 + 0.01(-0.019404) \approx 0.009806.\end{aligned} \tag{10.65}$$

Interpretation:
The proportion of A–users decreases further from 0.01 to about 0.0098. This shows that, starting near the pure B population, the dynamics move even closer to $(0, 1)$, confirming that the all–B population is evolutionarily stable in this model.

The above answers the question, here is some code to confirm the calculations.

```
h = 0.01
initial_x_1 = 0.01
initial_x_1 + h * x_1_dash.subs({x_1: initial_x_1})
```

```
0.00980596000000000
```

- (b) Initial population $x = (0.49, 0.51)$

Here $x_1^{(0)} = 0.49$:

$$\begin{aligned}\dot{x}_1\big|_{0.49} &= -2 \cdot 0.49 \cdot (2 \cdot 0.49 - 1) \cdot (0.49 - 1) \\ &= -2 \cdot 0.49 \cdot (-0.02) \cdot (-0.51) \\ &\approx -0.009996.\end{aligned} \tag{10.66}$$

So

$$\begin{aligned}x_1^{(1)} \\ = 0.49 + 0.01(-0.009996) \approx 0.489900.\end{aligned} \tag{10.67}$$

Interpretation:
Starting slightly below the mixed state $\left(\frac{1}{2}, \frac{1}{2}\right)$, the proportion of A–users decreases further (from 0.49 to about 0.4899). A small perturbation away from the mixed state does not return; instead it is pushed further away. This indicates that the mixed state is **not** evolutionarily stable.

The above answers the question, here is some code to confirm the calculations.

```
h = 0.01
initial_x_1 = 0.49
initial_x_1 + h * x_1_dash.subs({x_1: initial_x_1})
```

```
0.489900040000000
```

- Initial population $x = (0.99, 0.01)$

Here $x_1^{(0)} = 0.99$:

$$\begin{aligned} \dot{x}_1\big|_{0.99} &= -2 \cdot 0.99 \cdot (2 \cdot 0.99 - 1) \cdot (0.99 - 1) \\ &= -2 \cdot 0.99 \cdot (0.98) \cdot (-0.01) \\ &\approx 0.019404. \end{aligned} \tag{10.68}$$

Thus

$$\begin{aligned} &x_1^{(1)} \\ &\quad = 0.99 + 0.01(0.019404) \approx 0.990194. \end{aligned} \tag{10.69}$$

Interpretation:
The proportion of A–users increases slightly from 0.99 to about 0.9902. Starting near the pure A population, the dynamics move even closer to $(1, 0)$, showing that the all–A population is also evolutionarily stable.

The above answers the question, here is some code to confirm the calculations.

```
h = 0.01
initial_x_1 = 0.99
initial_x_1 + h * x_1_dash.subs({x_1: initial_x_1})
```

```
0.990194040000000
```

Overall, the dynamics push the population towards one of the two pure coordination states, and away from the mixed state.

8. We now let the strength of the coordination benefit be a parameter $a \neq 0$:

$$f_A(x) = 1 + ax_1, \qquad f_B(x) = 1 + a(1 - x_1). \tag{10.70}$$

- (a) Replicator dynamics for general a

The average fitness is

$$\begin{aligned} \phi(x) &= x_1 f_A(x) + (1 - x_1) f_B(x) \\ &= x_1(1 + ax_1) + (1 - x_1)\ \big(\ 1 + a(1 - x_1)\ \big) \\ &= 2ax_1^2 - 2ax_1 + a + 1. \end{aligned} \tag{10.71}$$

Then

$$\begin{aligned} f_A(x) - \phi(x) &= (1 + ax_1) - (2ax_1^2 - 2ax_1 + a + 1) \\ &= a(-2x_1^2 + 3x_1 - 1) \\ &= -a(2x_1 - 1)(x_1 - 1). \end{aligned} \tag{10.72}$$

Thus the replicator dynamics is

$$\begin{aligned}\frac{dx_1}{dt} &\\ &= x_1 \left(f_A(x) - \phi(x) \right) \\ &= -a\, x_1(2x_1 - 1)(x_1 - 1).\end{aligned} \tag{10.73}$$

The fixed points satisfy $\frac{dx_1}{dt} = 0$, so again

$$x_1 \in \left\{0, \frac{1}{2}, 1\right\}. \tag{10.74}$$

To determine stability, we can consider all potential post entry populations or we can equivalently examine the sign of $\dot{x}_1$:

First let us consider the case $a > 0$:

- For $0 < x_1 < \frac{1}{2}$:

$x_1 > 0, 2x_1 - 1 < 0, x_1 - 1 < 0$ so
$-ax_1(2x_1 - 1)(x_1 - 1) < 0$.
Thus for the post entry population $x_1 = 1/2 - \varepsilon\ x_1(t)$ decreases and the flow moves away from $x_1 = \frac{1}{2}$ towards $x_1 = 0$.

- For $\frac{1}{2} < x_1 < 1$:

$x_1 > 0, 2x_1 - 1 > 0, x_1 - 1 < 0$ so
$-ax_1(2x_1 - 1)(x_1 - 1) > 0$.
Thus for the post entry population $x_1 = 1/2 + \varepsilon\ x_1(t)$ increases and the flow moves away from $x_1 = \frac{1}{2}$ towards $x_1 = 1$.

Hence:

- $x_1 = \frac{1}{2}$ is **unstable**;
- $x_1 = 0$ and $x_1 = 1$ are **evolutionary stable**.

Now let us consider the case $a < 0$:

- For $0 < x_1 < \frac{1}{2}$:

$x_1 > 0, 2x_1 - 1 < 0, x_1 - 1 < 0$ so
$-ax_1(2x_1 - 1)(x_1 - 1) > 0$.
Thus for the post entry population $x_1 = 1/2 - \varepsilon\ x_1(t)$ increases and the flow moves towards $x_1 = \frac{1}{2}$.

- For $\frac{1}{2} < x_1 < 1$:

$x_1 > 0, 2x_1 - 1 > 0, x_1 - 1 < 0$ so
$-ax_1(2x_1 - 1)(x_1 - 1) < 0$.
Thus for the post entry population $x_1 = 1/2 + \varepsilon\ x_1(t)$ decreases and the flow moves towards $x_1 = \frac{1}{2}$.

If $a = 0$ the derivative is 0 and thus all populations are stable.

Hence:

- $x_1 = \frac{1}{2}$ is **evolutionary stable**;
- $x_1 = 0$ and $x_1 = 1$ are **unstable**.

So $x^* = (1/2, 1/2)$ is an evolutionary stable strategy if and only if $a < 0$: the network effect is bad. There will only ever be an emergent population using both apps when the network effect is in fact negative.

The above answers the question. Here is some code to confirm the calculations and illustrate the sign of the derivative.

```
a = sym.Symbol("a")
f_A = 1 + a * x_1
f_B = 1 + a * (1 - x_1)
phi = x_1 * f_A + (1 - x_1) * f_B
sym.simplify(phi)
```

```
2*a*x_1**2 - 2*a*x_1 + a + 1
```

```
x_1_dash = x_1 * (f_A - phi)
sym.simplify(x_1_dash)
```

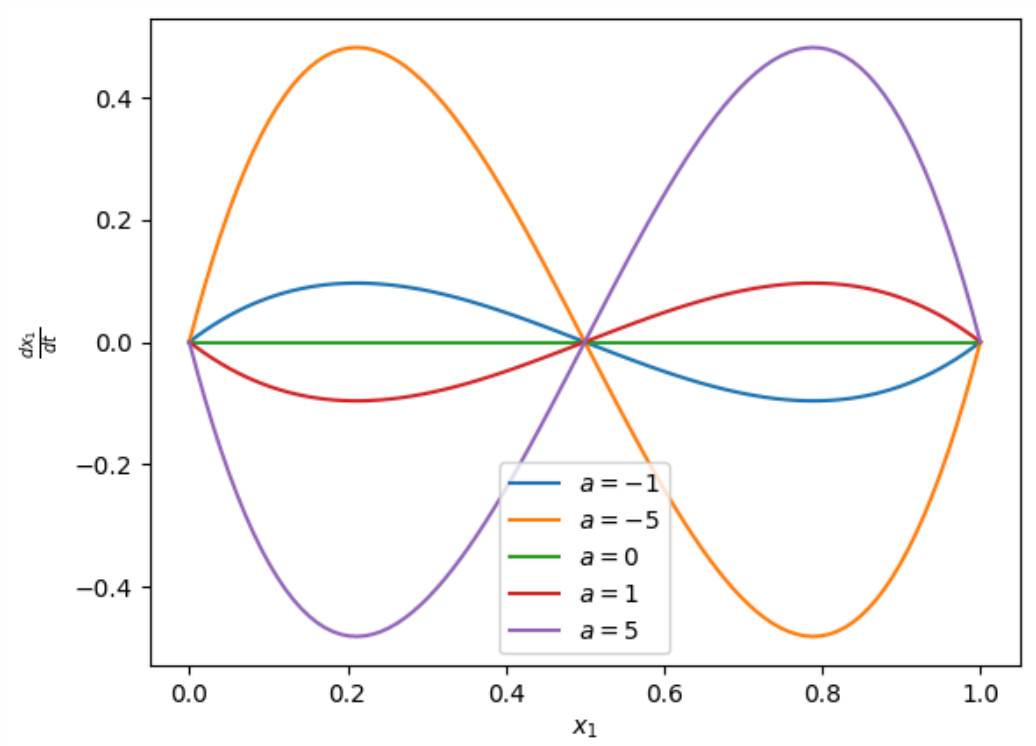
```
a*x_1*(-2*x_1**2 + 3*x_1 - 1)
```

```
x_1_dash = x_1 * (f_A - phi)
sym.solveset(x_1_dash, x_1)
```

```
{0, 1/2, 1}
```

```
import matplotlib.pyplot as plt
import numpy as np

x_1 = sym.Symbol("x_1")

a = sym.Symbol("a")
f_A = 1 + a * x_1
f_B = 1 + a * (1 - x_1)
phi = x_1 * f_A + (1 - x_1) * f_B
x_1_dash = x_1 * (f_A - phi)
x_1_values = np.linspace(0, 1, 100)

plt.figure()
plt.plot(x_1_values, [x_1_dash.subs({x_1: x_value, a: -1}) for x_value in
x_1_values], label="$a=-1$")
plt.plot(x_1_values, [x_1_dash.subs({x_1: x_value, a: -5}) for x_value in
x_1_values], label="$a=-5$")
plt.plot(x_1_values, [x_1_dash.subs({x_1: x_value, a: 0}) for x_value in
x_1_values], label="$a=0$")
plt.plot(x_1_values, [x_1_dash.subs({x_1: x_value, a: 1}) for x_value in
x_1_values], label="$a=1$")
plt.plot(x_1_values, [x_1_dash.subs({x_1: x_value, a: 5}) for x_value in
x_1_values], label="$a=5$")
plt.legend()
plt.xlabel("$x_1$")
plt.ylabel(r"$\frac{dx_1}{dt}$")
```

```
Text(0, 0.5, '$\\frac{dx_1}{dt}$')
```



11. Moran Process

In finite populations, random drift can overturn selection: a rare mutant strategy may take hold by chance even if it offers no advantage. This chapter introduces the Moran process to study such fixation events, analysing the probability that a single mutant takes over or disappears.

11.1 Motivating Example: Everyone's citing the preprint

In a graduate student reading group, everyone cites a well-established textbook in their essays. One student, though, starts citing a **recent preprint** they found on arXiv.

> "It's got a better proof of the key result, and it's open access."

Each week:

- A student is admired for their choice of citation (fitness $\propto$ novelty, clarity, or style).
- Another student, chosen at random, updates their references to match.

Over time, the group begins to shift its citation culture. The **preprint might become canonical**, or the students might return to citing the traditional text.

The more students who cite the preprint, the more attractive it becomes to others: shared references lead to easier discussion, common assumptions, and social reinforcement. In this way, the **fitness** of citing the preprint is not fixed; it **depends on the current citation habits** of the group. This makes the process **frequency-dependent**, just as in evolutionary game dynamics where payoffs arise from interaction with others.

To model this interaction explicitly, consider the following symmetric game. Each player chooses whether to cite the **textbook** (T) or the **preprint** (P). The row and column player payoffs are given by:

$$M_r = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \tag{11.1}$$

If both students cite the textbook, they align well and receive the highest payoff of 3. If both cite the preprint, they still coordinate and get a payoff of 2, slightly lower, but still beneficial.

If one cites the textbook while the other cites the preprint, there is a mismatch: the **preprint-citer** still gets some benefit (payoff 1) from its clarity and openness, but the textbook-citer gains nothing (payoff 0) from the mismatch.

Note here that the group is small and so the infinite population assumption of Replicator Dynamics does not apply: the topic of this chapter is the Moran Process a model suited for exactly this purpose.

> **Note**
>
> The Moran process was originally developed in population genetics to model **genetic drift**, the random fluctuation of gene frequencies in small biological populations. For the biological context and the relationship between drift and selection, see The Biological Origins of Evolutionary Game Theory.

11.2 Theory

11.2.1 Definition: Moran Process

First defined in (Moran, 1958), the Moran process assumes a constant population of N individuals which can be of m different types. There exists a fitness function $f : \{1, ..., m\} \times \{1, ..., m\}^N \to$

$\mathbb{R}$ that maps each individual to a numeric fitness value which is dependent on the types of the individuals in the population.

The process is defined as follows. At each step:

1. Every individual k has their fitness f_k calculated.
2. An individual is randomly selected for copying. This selection is done proportional to their fitness $f_k(v)$. Thus, the probability of selecting individual k for copying is given by:

$$\frac{f_k(v)}{\sum_{h=1}^{N} f_h(v)} \tag{11.2}$$

1. An individual is selected for removal. This selection is done uniformly at random. Thus, the probability of selecting individual i for removal is:

$$\frac{1}{N} \tag{11.3}$$

1. An individual of the same type as the individual selected for copying is introduced to the population.
2. The individual selected for removal is removed.

The process is repeated until there is only one type of individual left in the population.

A common representation of the fitness function f is to use a game. In this setting, the fitness of an individual of type i is:

$$f_i(v) = (v_i - 1)A_{ii} + \sum_{j \neq i, j=1}^{N} v_j A_{ij} \tag{11.4}$$

11.2.1.1 Example: Selection Probabilities for citation behaviour.

For the Motivating Example: Everyone's citing the preprint let us consider the situation with N individuals in the reading group: 3 cite the text book (T) and 1 cites the preprint (P). This gives a total number of 5 different populations. Table 11.1 gives the different selection probabilities for each population.

(v_T, v_P)	f_T	f_P	Prob copy T	Prob copy P	Prob remove T	Prob remove P
$(4,0)$	$3 \cdot 3 = 9$	–	1	0	1	0
$(3,1)$	$2 \cdot 3 + 1 \cdot 0 = 6$	$3 \cdot 1 + 0 \cdot 2 = 3$	$\frac{3 \cdot 6}{3 \cdot 6 + 1 \cdot 3} = \frac{6}{7}$	$\frac{1}{7}$	$\frac{3}{4}$	$\frac{1}{4}$
$(2,2)$	$1 \cdot 3 + 2 \cdot 0 = 3$	$2 \cdot 1 + 1 \cdot 2 = 4$	$\frac{2 \cdot 3}{2 \cdot 3 + 2 \cdot 4} = \frac{3}{7}$	$\frac{4}{7}$	$\frac{1}{2}$	$\frac{1}{2}$
$(1,3)$	$0 \cdot 3 + 3 \cdot 0 = 0$	$1 \cdot 1 + 2 \cdot 2 = 5$	$\frac{1 \cdot 0}{1 \cdot 0 + 3 \cdot 5} = 0$	1	$\frac{1}{4}$	$\frac{3}{4}$
$(0,4)$	–	$0 \cdot 1 + 3 \cdot 2 = 6$	0	1	0	1

Table 11.1: Selection probabilities for citation behaviour

11.2.2 Definition: The Fixation Probability

The fixation probability of a given type in a Moran process is the probability that the population eventually becomes composed entirely of individuals of that type.

In the case of a finite population of size N with **two** types: a resident type and a mutant type. Suppose the process begins with i individuals of the mutant type and $N - i$ residents.

Let ρ_i denote the fixation probability of the mutant type starting from i individuals. Then:

- $\rho_0 = 0$, since no mutants exist.
- $\rho_N = 1$, since all individuals are mutants.
- For $0 < i < N$, ρ_i gives the probability that the mutant type eventually fixates (i.e., reaches frequency N), assuming the dynamics follow the Moran process.

In practice fixation probabilities correspond to absorption probabilities of an underlying absorbing Markov chain.

Then, the transition probabilities for the underling Markov chain are:

$$P_{i \to i+1} = (\text{Prob copy mutant}) \cdot (\text{Prob remove resident}) \tag{11.5}$$

and

$$P_{i \to i-1} = (\text{Prob copy resident}) \cdot (\text{Prob remove mutant}) \tag{11.6}$$

Finally:

$$P_{i \to i} = 1 - P_{i \to i+1} - P_{i \to i-1} \tag{11.7}$$

11.2.2.1 Example: Fixation of citation behaviour as an absorbing Markov chain

For given N the Motivating Example: Everyone's citing the preprint the underlying absorbing Markov chain has a state space that can be indexed by i the number of individuals that cite the preprint.

For $N = 4$, we can write down the probability of going from state i to state j: p_{ij} using Table 11.1:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ (6/7) \cdot (1/4) & P_{11} & (1/7) \cdot (3/4) & 0 & 0 \\ 0 & (3/7) \cdot (1/2) & P_{22} & (4/7) \cdot (1/2) & 0 \\ 0 & 0 & 0 \cdot (3/4) & P_{33} & (1) \cdot (1/4) \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{using the selection probabilities} \tag{11.8}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3/14 & 19/28 & 3/28 & 0 & 0 \\ 0 & 3/14 & 1/2 & 2/7 & 0 \\ 0 & 0 & 0 & 3/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{using } P_{i \to i} = 1 - P_{i \to i+1} - P_{i \to i-1}$$

This is an absorbing Markov chain which, by reordering of states, we can write in the canonical form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{11.9}$$

with:

$$Q = \begin{pmatrix} 19/28 & 3/28 & 0 \\ 3/14 & 1/2 & 2/7 \\ 0 & 0 & 3/4 \end{pmatrix} \tag{11.10}$$

and

$$R = \begin{pmatrix} 3/14 & 0 \\ 0 & 0 \\ 0 & 1/4 \end{pmatrix} \tag{11.11}$$

thus we can calculate the fundamental matrix:

$$\begin{aligned} N &= \begin{pmatrix} 9/28 & -3/28 & 0 \\ -3/14 & 1/2 & -2/7 \\ 0 & 0 & 1/4 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 98/27 & 7/9 & 8/9 \\ 14/9 & 7/3 & 8/3 \\ 0 & 0 & 4 \end{pmatrix} \end{aligned} \tag{11.12}$$

We omit the calculation of the inverse which can be obtained using Gauss-Jordan elimination or any other approach.

We can now compute the absorption probability matrix:

$$B = NR = \begin{pmatrix} 7/9 & 2/9 \\ 1/3 & 2/3 \\ 0 & 1 \end{pmatrix} \tag{11.13}$$

Thus if a single mutant, or in our case a single individual starts citing the pre print: there is $2/9$ chance that the entire reading group starts citing the pre print over time.

$$\rho_1 = \frac{2}{9} \tag{11.14}$$

11.2.3 Theorem: The fixation probabilities in populations of two types

Given a Moran process in a population with two types as defined in Definition: The Fixation Probability, the fixation probability ρ_i is given by:

$$\rho_i = \frac{1 + \sum_{j=1}^{i-1} \prod_{k=1}^{j} \gamma_k}{1 + \sum_{j=1}^{N-1} \prod_{k=1}^{j} \gamma_k} \tag{11.15}$$

where:

$$\gamma_k = \frac{f_R(k)}{f_M(k)} \tag{11.16}$$

Proof:

For the underlying absorbing Markov chain we have:

$$\begin{aligned} p_{i,i+1}\rho_{i+1} &= -p_{i,i-1}\rho_{i-1} + \rho_i(1 - p_{ii}) \\ p_{i,i+1}\rho_{i+1} &= p_{i,i-1}(\rho_i - \rho_{i-1}) + \rho_i p_{i,i+1} \\ \rho_{i+1} - \rho_i &= \frac{p_{i,i-1}}{p_{i,i+1}}(\rho_i - \rho_{i-1}) = \gamma_i(\rho_i - \rho_{i-1}) \end{aligned} \tag{11.17}$$

with:

$$
\begin{aligned}
\gamma_i &= \frac{p_{i,i-1}}{p_{i,i+1}} \\
&= \frac{\frac{(N-i)f_R(i)}{if_M(i)+(N-i)f_R(i)}\frac{i}{N}}{\frac{if_M(i)}{if_M(i)+(N-i)f_R(i)}\frac{N-i}{N}} \\
&= \frac{(N-i)f_R(i)}{if_M(i)+(N-i)f_R(i)}\frac{i}{N}\frac{if_M(i)+(N-i)f_R(i)}{if_M(i)}\frac{N}{N-i} \\
&= \frac{(N-i)f_R(i)i}{if_M(i)(N-i)} \\
&= \frac{f_R(i)}{f_M(i)}
\end{aligned}
\tag{11.18}
$$

We observe that:

$$
\begin{aligned}
\rho_2 - \rho_1 &= \gamma_1(\rho_1 - \rho_0) = \gamma_1\rho_1 \\
\rho_3 - \rho_2 &= \gamma_2(\rho_2 - \rho_1) = \gamma_2\gamma_1\rho_1 \\
\rho_4 - \rho_3 &= \gamma_3(\rho_3 - \rho_2) = \gamma_3\gamma_2\gamma_1\rho_1 \\
&\vdots \\
\rho_{i+1} - \rho_i &= \gamma_i(\rho_i - \rho_{i-1}) = \prod_{k=1}^{i}\gamma_k\rho_1 \\
&\vdots \\
\rho_N - \rho_{N-1} &= \gamma_{N-1}(\rho_{N-1} - \rho_{N-2}) = \prod_{k=1}^{N-1}\gamma_k\rho_1
\end{aligned}
\tag{11.19}
$$

thus we have:

$$
\rho_i = \sum_{j=0}^{i-1}\rho_{j+1} - \rho_j = \left(1 + \sum_{j=1}^{i-1}\prod_{k=1}^{j}\gamma_k\right)\rho_1 \tag{11.20}
$$

solving the following equation to obtain ρ_1 gives the required result.

$$
\rho_N = 1 = \left(1 + \sum_{j=1}^{N-1}\prod_{k=1}^{j}\gamma_k\right)\rho_1 \tag{11.21}
$$

11.2.3.1 Example: Direct calculation of fixation of citation behaviour

For given N the fixation probabilities of Motivating Example: Everyone's citing the preprint can be found directly using (11.15).

For $N = 4$, recalling that $R = T$ and $M = P$, we can write down the values of γ_i using Table 11.1:

$$
\begin{aligned}
\gamma_1 &= \frac{f_T(1)}{f_P(1)} = \frac{6}{3} = 2 \\
\gamma_2 &= \frac{f_T(2)}{f_P(2)} = \frac{3}{4} \\
\gamma_3 &= \frac{f_T(3)}{f_P(3)} = \frac{0}{5}
\end{aligned}
\tag{11.22}
$$

This gives:

$$\begin{aligned}\rho_1 &= \frac{1}{1+\sum_{j=1}^{3}\prod_{k=1}^{j}\gamma_k} \\ &= \frac{1}{1+\prod_{k=1}^{1}\gamma_k+\prod_{k=1}^{2}\gamma_k+\prod_{k=1}^{3}\gamma_k} \\ &= \frac{1}{1+\gamma_1+\gamma_1\gamma_2+\gamma_1\gamma_2\gamma_3} \\ &= \frac{1}{1+2+\frac{2\cdot3}{4}+\frac{2\cdot3\cdot0}{4\cdot5}} \\ &= \frac{1}{1+2+\frac{6}{4}} = \frac{2}{9}\end{aligned} \tag{11.23}$$

as calculated Example: Fixation of citation behaviour as an absorbing Markov chain.

11.2.4 Selection Strength

The fitness function in the Moran process need not equal the raw game payoff directly. A parameter $w \in [0,1]$, the **intensity of selection** (or **selection strength**), controls how strongly payoffs influence reproductive success. The fitness of type i under selection intensity w is:

$$f_i = 1 - w + w\,\pi_i \tag{11.24}$$

where π_i is the expected payoff of type i against the current population.

11.2.5 Definition: Relative fitness and the constant-fitness Moran process

A **constant-fitness Moran process** is one in which each type has a fitness that does not depend on the composition of the population. With two types, a resident of fitness f_R and a mutant of fitness f_M, the **relative fitness** of the mutant is the ratio:

$$r = \frac{f_M}{f_R} \tag{11.25}$$

It is conventional to normalise the resident fitness to $f_R = 1$, so that the mutant has fitness $f_M = r$ and r measures how much fitter the mutant is than the resident. Thus $r > 1$ corresponds to an advantageous mutant, $r < 1$ to a disadvantageous one, and $r = 1$ to a neutral mutant.

For a constant-fitness process every ratio $\gamma_k = f_R(k)/f_M(k) = r^{-1}$ is the same, so the products in (11.15) become powers of r^{-1} and the fixation probability starting from i mutants has the closed form:

$$\rho_i = \frac{1 - r^{-i}}{1 - r^{-N}}, \qquad r \neq 1 \tag{11.26}$$

In particular, a single mutant fixates with probability:

$$\rho_1 = \frac{1 - r^{-1}}{1 - r^{-N}} \tag{11.27}$$

The neutral case $r = 1$ is recovered as a limit below, and gives $\rho_1 = 1/N$.

11.2.6 Definition: Neutral Drift

A Moran process exhibits **neutral drift** when $w = 0$, so that every individual has fitness 1 regardless of strategy. The fixation probability of a single mutant is:

$$\rho_1^{\text{neutral}} = \frac{1}{N} \tag{11.28}$$

This follows directly from (11.15): with all fitnesses equal, $\gamma_k = 1$ for all k, and the formula reduces to $1/N$. A mutant fixates with exactly the same probability as any randomly chosen individual would be expected to, so chance alone determines the outcome.

11.2.7 Definition: Strong Selection

A Moran process exhibits **strong selection** when $w = 1$, so that the fitness of each type equals its raw payoff:

$$f_i = \pi_i \tag{11.29}$$

The outcome depends entirely on which strategy earns higher payoffs against the current population composition. Payoff differences are felt at full strength at every step of the process.

11.2.8 Definition: Weak Selection

A Moran process operates under **weak selection** when $0 < w \ll 1$, so that payoff differences introduce only a small perturbation to the neutral baseline. The fixation probability admits a first-order expansion in w:

$$\rho_1 = \frac{1}{N} + w \cdot c + O(w^2) \tag{11.30}$$

for a constant c that depends on the payoff matrix and population size N. A mutant is **favoured by selection** if $\rho_1 > 1/N$.

The weak-selection regime is widely used because it permits analytical results for arbitrary games: whether a mutant is favoured reduces to a linear condition on the payoff matrix entries alone (Nowak, 2006).

Note

The parameterisation $f_i = 1 - w + w\pi_i$ ensures fitness remains positive for small w even when π_i can be negative, and reduces continuously to neutral drift as $w \to 0$. It is equivalent to the constant-fitness formulation $W_A = 1 + w(\pi_A - 1)$, $W_B = 1$, used in the fitness recurrence of Fitness: Reproductive Success as Payoff.

11.3 Exercises

Exercise 11.1:

A Moran process with neutral drift is when: $f_k v = C$ for all k for all v for some constant C. In other words: a Moran process with neutral drift is a Moran process where the fitness of all types for all populations is the same.

For a population with 2 types:

1. Describe the transition probabilities for the Moran process with neutral drift.

2. Obtain the transition probability matrix for the Moran process with neutral drift with $N = 4$ individuals.
3. Obtain the general formula for ρ_1 for a Moran process with neutral drift for general N.

Exercise 11.2:

For the following games, assuming the mutant is of the second type, obtain the fixation probability ρ_1 for $N = 4$:

1. $M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$
2. $M = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$

Exercise 11.3:

Consider the game $M = \begin{pmatrix} r & 1 \\ 1 & 1 \end{pmatrix}$ for $r > 1$ and N, assuming the mutant is of the second type, obtain ρ_1 as a function of r. How does r effect the chance of fixation?

Exercise 11.4:

Consider the constant-fitness Moran process in which a mutant type has fitness $f_M = 1 - w + wr$ and the resident has fitness $f_R = 1 - w + w \cdot 1 = 1$, where $r > 0$ and $w \in [0, 1]$ is the selection intensity. There are N individuals in total.

1. Show that $\gamma_k = f_R/f_M$ is independent of k, and write down its value as a function of w and r.
2. Using (11.15), derive a closed-form expression for ρ_1 in terms of γ.
3. Show that as $w \to 0$ (neutral drift), $\rho_1 \to 1/N$.
4. For $r > 1$ (the mutant has a higher raw payoff than the resident), show that $\rho_1 > 1/N$ for all $w > 0$: the mutant is favoured by selection at every positive selection intensity.
5. Interpret the two limits $w \to 0$ (weak selection) and $w \to 1$ (strong selection) biologically. In which regime does random drift dominate, and in which does payoff advantage dominate?

Exercise 11.5:

Consider the following matrix:

$$M = \begin{pmatrix} 3a & a \\ 2a & 2 \end{pmatrix}, \qquad a > 0. \tag{11.31}$$

1. Show that for all $a > 0$ the game defined by M, M^T is **not** a Prisoners' Dilemma.
2. Find all **Nash equilibria** of the game as a function of a.
3. Now consider a **two-type Moran process** with a population of size N.

Type 1 individuals play row 1; type 2 individuals play row 2.

The fitness of each type is given by:

$$f_1(i) = \left(\frac{(i-1)M_{11} + (N-i)M_{12}}{N-1} \right), \tag{11.32}$$

$$f_2(i) = \left(\frac{i\,M_{21} + (N-i-1)M_{22}}{N-1} \right), \tag{11.33}$$

where i is the number of type-1 individuals.

Using the standard formula:

$$\rho_i = \frac{1 + \sum_{j=1}^{i-1} \prod_{k=1}^{j} \gamma_k}{1 + \sum_{j=1}^{N-1} \prod_{k=1}^{j} \gamma_k}, \qquad \gamma_k = \frac{f_2(k)}{f_1(k)}, \tag{11.34}$$

4. Compute explicitly the fixation probability of a **single mutant** (ρ_1) for

$N \in \{2, 3, 4\}$.

4. For $N \in \{2, 3, 4\}$, analyse the fixation probability ρ_1 in the two limits:
5. $a \to 0$,
6. $a \to \infty$.

Explain the evolutionary intuition behind these two limiting behaviours and relate them to the role of **fitness amplification** in frequency-dependent selection.

11.4 Programming

11.4.1 Using Nashpy to simulate a Moran process

Nashpy has functionality to simulate a single Moran process. Let us create a 3 by 3 game (for a population with 3 types) and an initial population.

```
import nashpy as nash
import numpy as np

M = np.array(
    (
        (2, 3, 1),
        (4, 1, 2),
        (1, 2, 5),
    )
)
game = nash.Game(M)
initial_population = np.array((0, 0, 0, 1, 1, 1, 2))
```

Note

The population above corresponds to 3 individuals of the first type, 3 of the second type and one of the third type. This is the format expected by Nashpy.

Now to run a Moran process, note that we seed the numpy pseudo-random number generator which is used by Nashpy:

```
np.random.seed(0)
populations = game.moran_process(initial_population=initial_population)
print(list(populations))
```

```
[array([0, 0, 0, 1, 1, 1, 2]), array([0, 0, 0, 1, 1, 1, 2]), array([0, 1, 0, 1,
1, 1, 2]), array([0, 1, 1, 1, 1, 1, 2]), array([0, 1, 1, 1, 1, 1, 1]), array([0,
1, 1, 1, 0, 1, 1]), array([0, 1, 1, 1, 0, 1, 1]), array([0, 1, 1, 1, 0, 1, 1]),
array([0, 1, 1, 1, 0, 0, 1]), array([0, 1, 1, 1, 0, 0, 1]), array([0, 1, 1, 0, 0,
0, 1]), array([0, 1, 1, 0, 0, 1, 1]), array([1, 1, 1, 0, 0, 1, 1]), array([1, 1,
1, 0, 0, 1, 1]), array([1, 1, 1, 0, 0, 1, 1]), array([1, 1, 1, 0, 0, 1, 1]),
array([1, 0, 1, 0, 0, 1, 1]), array([0, 0, 1, 0, 0, 1, 1]), array([0, 0, 1, 0, 0,
1, 1]), array([0, 0, 1, 0, 0, 1, 0]), array([0, 0, 1, 0, 1, 1, 0]), array([1, 0,
1, 0, 1, 1, 0]), array([1, 0, 1, 0, 1, 1, 0]), array([1, 0, 1, 0, 1, 1, 0]),
array([1, 0, 1, 0, 1, 1, 0]), array([1, 0, 1, 0, 1, 1, 0]), array([1, 1, 1, 0, 1,
1, 0]), array([1, 1, 1, 0, 1, 1, 0]), array([1, 1, 1, 0, 1, 1, 0]), array([1, 1,
1, 0, 1, 1, 1]), array([1, 1, 1, 0, 1, 1, 1]), array([1, 1, 1, 0, 1, 1, 1]),
array([1, 1, 1, 0, 1, 1, 1]), array([1, 1, 1, 0, 0, 1, 1]), array([1, 1, 1, 0, 1,
1, 1]), array([1, 1, 1, 0, 1, 1, 1]), array([1, 1, 1, 0, 1, 1, 0]), array([1, 1,
1, 0, 1, 1, 0]), array([1, 1, 1, 0, 1, 1, 0]), array([1, 1, 1, 0, 1, 1, 0]),
array([1, 1, 1, 0, 1, 1, 0]), array([1, 1, 1, 0, 1, 0, 0]), array([1, 1, 1, 0, 1,
0, 0]), array([1, 1, 0, 0, 1, 0, 0]), array([1, 1, 0, 0, 1, 0, 0]), array([1, 1,
0, 0, 1, 0, 0]), array([1, 1, 0, 0, 1, 1, 0]), array([1, 1, 0, 0, 1, 1, 0]),
array([1, 1, 0, 0, 1, 1, 0]), array([1, 1, 0, 1, 1, 1, 0]), array([1, 1, 1, 1, 1,
1, 0]), array([1, 1, 1, 1, 1, 1, 0]), array([1, 1, 1, 1, 1, 1, 0]), array([1, 1,
1, 1, 1, 1, 0]), array([1, 1, 1, 1, 1, 1, 0]), array([1, 1, 0, 1, 1, 1, 0]),
array([1, 1, 1, 1, 1, 1, 0]), array([1, 1, 1, 1, 1, 1, 0]), array([1, 1, 1, 1, 1,
1, 0]), array([1, 1, 1, 1, 1, 1, 1])]
```

11.4.2 Using Nashpy to approximate fixation probabilities

Nashpy can be directly used to approximate the fixation probabilities by repeated a large number of Moran processes:

```
M = np.array(
    (
        (3, 0),
        (1, 2),
    )
)
game = nash.Game(M)
initial_population = np.array((0, 0, 0, 1))
fixation_probabilities = game.fixation_probabilities(
    initial_population=initial_population, repetitions=2_000
)
print(f"Fixation probabilities: {fixation_probabilities}")
```

```
Fixation probabilities: {(np.int64(0), np.int64(0), np.int64(0), np.int64(0)):
0.789, (np.int64(1), np.int64(1), np.int64(1), np.int64(1)): 0.211}
```

This shows that the final population with only 1s in it occurs $2/9 \approx 0.22$ of the time.

11.5 Notable Research

11.5.1 Notable Research in the Moran Process and Population Genetics

The Moran process was first introduced in (Moran, 1958), but it was not the first major model in population genetics. Foundational theoretical work by Ronald Fisher (Fisher, 1930), J.B.S. Haldane (Haldane, 1927; 1932), and Sewall Wright (Wright, 1931) laid the mathematical groundwork for understanding evolution, focusing on selection and genetic drift in both infinite and finite populations. These early models often used diffusion approximations or the discrete-generation Wright-Fisher model. In contrast, the Moran process provided a continuous-time, discrete-space alternative that allows exact calculation of fixation probabilities and times. For example, Antal and Scheuring (Antal & Scheuring, 2006) derived precise analytical results within this framework.

The Moran process has become indispensable in evolutionary game theory, where individual fitness depends on strategic interactions. This is central to the theory of evolutionarily stable strategies (Taylor & Jonker, 1978). It has been especially influential in studying social dilemmas, such as the evolution of cooperation. Traulsen and Nowak (Traulsen & Nowak, 2006) showed how cooperation can be favoured in finite populations, while Knight (Knight et al., 2018) explored how self-recognition algorithms can emerge through such dilemmas under the Moran process.

The process is also crucial for analysing the role of population structure. A notable extension is the Moran process on graphs, where individuals interact only with their neighbors. This framework was first proposed by Lieberman, Hauert, and Nowak (Lieberman et al., 2005) and further refined by Ohtsuki, Pacheco, and Nowak (Ohtsuki et al., 2007). The `Nashpy` library (Knight & Campbell, 2018) can be used to simulate Moran processes on such networks.

A final, and remarkable, result is the one proved by Traulsen, Claussen, and Hauert (Traulsen et al., 2005): in the limit of large population size, the Moran process converges to the replicator dynamics equation.

11.6 Conclusion

The Moran process offers a foundational framework for understanding how strategies evolve in finite populations. Like the replicator dynamics equation, it links fitness to the growth or decline of types over time, but with a critical distinction: it captures the inherent stochasticity of small populations.

The central result is the fixation probability formula: by mapping the process to an absorbing Markov chain, exact analysis becomes tractable for two-type populations.

The key concepts covered in this chapter are summarized in Table 11.2.

Concept	Description
Moran process	A stochastic model of evolution in finite populations
Copy selection probability	Probability of choosing an individual for reproduction, proportional to fitness
Removal selection probability	Probability of choosing an individual for removal, uniform across the population
Absorbing state	A population state in which all individuals are of a single type
Fixation probability	Probability that a given type eventually takes over the population

Table 11.2: Summary of key concepts in the Moran process

Important

In a finite population the Moran Process will terminate with non zero probability in a state with all individuals being of the same type.

The fixation probability quantifies the likelihood that a particular type ultimately dominates the population.

11.7 Solutions

Solution 11.1: Solution to Exercise 1

1. Under neutral drift every individual has the same fitness C, so selection for copying is uniform. In a state with i mutants and $N-i$ residents a mutant is copied with probability i/N and a resident with probability $(N-i)/N$; removal is uniform, so a resident is removed with probability $(N-i)/N$ and a mutant with probability i/N. Hence

$$P_{i\to i+1} = \frac{i}{N}\cdot\frac{N-i}{N} = \frac{i(N-i)}{N^2}, \qquad P_{i\to i-1} = \frac{N-i}{N}\cdot\frac{i}{N} = \frac{i(N-i)}{N^2}, \tag{11.35}$$

with $P_{i\to i} = 1 - 2i(N-i)/N^2$. The up and down probabilities are equal, so the process is a symmetric random walk on $\{0,1,\ldots,N\}$ with the two endpoints absorbing.

1. For $N=4$ we have $i(N-i)/16$ equal to $3/16$, $4/16 = 1/4$ and $3/16$ for $i=1,2,3$. Indexing the states by the number of mutants $i \in \{0,1,2,3,4\}$,

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3/16 & 5/8 & 3/16 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 3/16 & 5/8 & 3/16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{11.36}$$

1. Since every $\gamma_k = f_R(k)/f_M(k) = C/C = 1$, the products in (11.15) are all 1 and

$$\rho_i = \frac{1+\sum_{j=1}^{i-1} 1}{1+\sum_{j=1}^{N-1} 1} = \frac{i}{N}, \qquad \text{so} \qquad \rho_1 = \frac{1}{N}. \tag{11.37}$$

A single neutral mutant fixates with probability $1/N$, exactly its initial share of the population.

Solution 11.2: Solution to Exercise 2

The mutant is the second type, so residents are of the first type. In a state with k mutants the fitnesses are $f_R(k) = (N-k-1)M_{11} + kM_{12}$ and $f_M(k) = (N-k)M_{21} + (k-1)M_{22}$, and $\gamma_k = f_R(k)/f_M(k)$. With $N=4$,

$$\rho_1 = \frac{1}{1+\gamma_1+\gamma_1\gamma_2+\gamma_1\gamma_2\gamma_3}. \tag{11.38}$$

1. For $M = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)$ every payoff equals 1, so $f_R(k) = (N-k-1)+k = N-1$ and $f_M(k) = (N-k)+(k-1) = N-1$. Thus $\gamma_k = 1$ for all k: this is neutral drift and $\rho_1 = 1/N = 1/4$.
2. For $M = \left(\begin{smallmatrix} 1 & 2 \\ 3 & 1 \end{smallmatrix}\right)$ we obtain $f_R(k) = (3-k)+2k = 3+k$ and $f_M(k) = 3(4-k)+(k-1) = 11-2k$, so $\gamma_k = (3+k)/(11-2k)$. Hence $\gamma_1 = 4/9$, $\gamma_2 = 5/7$ and $\gamma_3 = 6/5$, giving

$$\rho_1 = \frac{1}{1 + \frac{4}{9} + \frac{4}{9}\cdot\frac{5}{7} + \frac{4}{9}\cdot\frac{5}{7}\cdot\frac{6}{5}} = \frac{1}{15/7} = \frac{7}{15}. \tag{11.39}$$

```
import sympy as sym

def fixation_probability(M, N):
    resident_fitness = lambda k: (N - k - 1) * M[0][0] + k * M[0][1]
    mutant_fitness = lambda k: (N - k) * M[1][0] + (k - 1) * M[1][1]
    gammas = [
        sym.Rational(resident_fitness(k), mutant_fitness(k)) for k in range(1,
N)
    ]
    total = 1
    product = 1
    for gamma in gammas:
        product *= gamma
        total += product
    return sym.simplify(1 / total)

print(fixation_probability([[1, 1], [1, 1]], 4))
print(fixation_probability([[1, 2], [3, 1]], 4))
```

```
1/4
7/15
```

Solution 11.3: Solution to Exercise 3

The mutant is the second type. Since the second row of $M = \begin{pmatrix} r & 1 \\ 1 & 1 \end{pmatrix}$ is constant, a mutant always earns 1 against any opponent, while residents earn r against each other and 1 against a mutant. In a state with k mutants,

$$f_R(k) = (N-k-1)r + k, \qquad f_M(k) = (N-k) + (k-1) = N-1, \tag{11.40}$$

so

$$\gamma_k = \frac{f_R(k)}{f_M(k)} = \frac{r(N-1-k)+k}{N-1}. \tag{11.41}$$

For $r > 1$ and $k < N-1$ we have $\gamma_k > 1$, while $\gamma_{N-1} = 1$; the residents are the fitter type and the products grow with r. The fixation probability is therefore

$$\rho_1 = \frac{1}{1 + \sum_{j=1}^{N-1} \prod_{k=1}^{j} \gamma_k}, \tag{11.42}$$

a decreasing function of r. Evaluating for small N:

- $N = 2$: $\gamma_1 = 1$ and $\rho_1 = 1/2$ for every r. With a single resident and a single mutant neither type has yet met its own kind, so the contest is neutral.
- $N = 3$: $\gamma_1 = (r+1)/2$ and $\gamma_2 = 1$, giving $\rho_1 = 1/(r+2)$.
- $N = 4$: $\gamma_1 = (2r+1)/3$, $\gamma_2 = (r+2)/3$ and $\gamma_3 = 1$, giving $\rho_1 = 9/\left(4(r+2)^2\right)$.

As $r \to 1$ each expression returns the neutral value $1/N$, and as $r \to \infty$ the fixation probability tends to 0 for $N \geq 3$: the fitter the residents, the less likely a mutant is to take over. Because the mutant's fitness is fixed at $N - 1$, the parameter r acts entirely through the residents.

```
import sympy as sym

r = sym.Symbol("r", positive=True)

def fixation_probability(M, N):
    resident_fitness = lambda k: (N - k - 1) * M[0][0] + k * M[0][1]
    mutant_fitness = lambda k: (N - k) * M[1][0] + (k - 1) * M[1][1]
    gammas = [resident_fitness(k) / mutant_fitness(k) for k in range(1, N)]
    total = 1
    product = 1
    for gamma in gammas:
        product *= gamma
        total += product
    return sym.simplify(1 / total)

for N in (2, 3, 4):
    rho_1 = fixation_probability([[r, 1], [1, 1]], N)
    print(N, rho_1, sym.limit(rho_1, r, sym.oo))
```

```
2 1/2 1/2
3 1/(r + 2) 0
```

```
4 9/(4*(r**2 + 4*r + 4)) 0
```

Solution 11.4: Solution to Exercise 4

1. With $f_M = 1 - w + wr$ and $f_R = 1$, the ratio is:

$$\gamma_k = \frac{f_R}{f_M} = \frac{1}{1 - w + wr} = \frac{1}{1 + w(r - 1)} \tag{11.43}$$

which is independent of k (constant-fitness Moran process).

2. Since $\gamma_k = \gamma$ for all k, the products simplify: $\prod_{k=1}^{j} \gamma_k = \gamma^j$. Thus:

$$\rho_1 = \frac{1}{1 + \sum_{j=1}^{N-1} \gamma^j} = \frac{1}{\sum_{j=0}^{N-1} \gamma^j} \tag{11.44}$$

For $\gamma \neq 1$ this is a geometric sum:

$$\rho_1 = \frac{1 - \gamma}{1 - \gamma^N} \tag{11.45}$$

3. As $w \to 0$, $\gamma \to 1$. Applying L'Hôpital's rule (or noting both numerator and denominator tend to 0):

$$\lim_{\gamma \to 1} \frac{1 - \gamma}{1 - \gamma^N} = \lim_{\gamma \to 1} \frac{-1}{-N\gamma^{N-1}} = \frac{1}{N} \tag{11.46}$$

So $\rho_1 \to 1/N$ as $w \to 0$, confirming the neutral drift result.

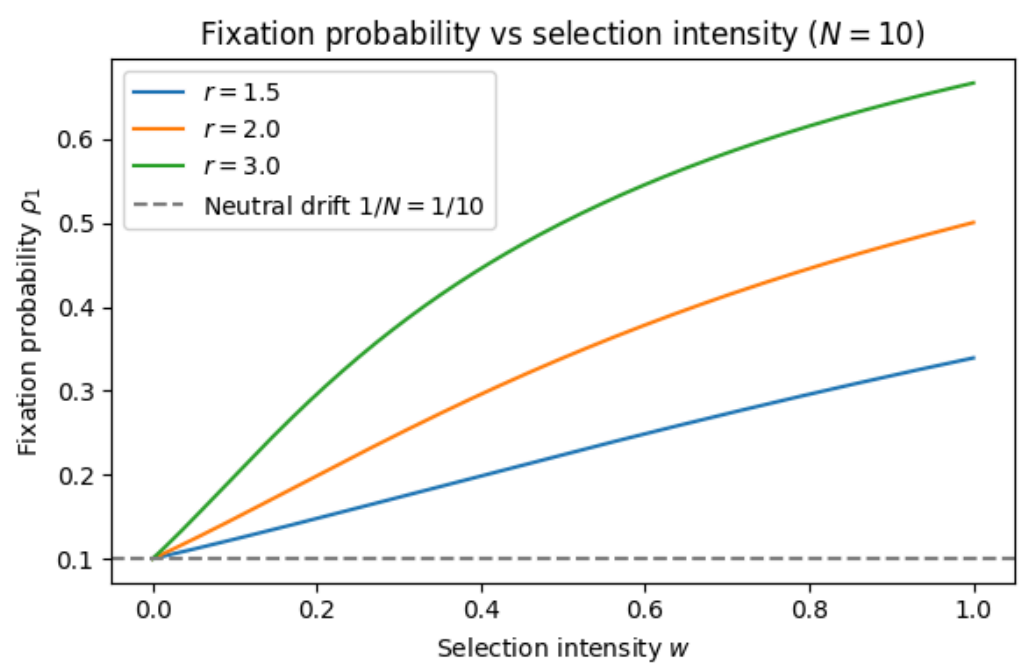
4. For $r > 1$ and $w > 0$, we have $\gamma = 1/(1 + w(r - 1)) < 1$. With $\gamma < 1$ the fixation probability is:

$$\rho_1 = \frac{1 - \gamma}{1 - \gamma^N} \tag{11.47}$$

We need to show $\rho_1 > 1/N$, i.e. $N(1 - \gamma) > 1 - \gamma^N$. Define $g(\gamma) = 1 - \gamma^N - N(1 - \gamma)$ for $\gamma \in (0, 1)$. Then $g(1) = 0$ and $g'(\gamma) = -N\gamma^{N-1} + N = N(1 - \gamma^{N-1}) > 0$ for $\gamma \in (0, 1)$. So g is increasing on $(0, 1)$ and $g(\gamma) < g(1) = 0$, meaning $1 - \gamma^N < N(1 - \gamma)$, i.e. $\rho_1 > 1/N$. □

5. Under **weak selection** ($w \approx 0$) the fitnesses of all individuals are nearly equal (all close to 1), so which individual is copied is nearly random. Random drift dominates: the mutant may fix or go extinct regardless of whether $r > 1$ or $r < 1$, and the fixation probability is close to $1/N$. Under **strong selection** ($w = 1$) fitness differences are maximal. If $r > 1$, the mutant is substantially fitter than the resident at every population state, and the fixation probability is maximised. In the extreme, as $r \to \infty$, $\gamma \to 0$ and $\rho_1 \to 1$: the mutant is almost certain to take over. Strong selection amplifies the effect of fitness differences; weak selection washes them out.

```
import numpy as np
import matplotlib.pyplot as plt

def rho_1(w, r, N):
    gamma = 1 / (1 + w * (r - 1))
    if abs(gamma - 1) < 1e-12:
        return 1 / N
    return (1 - gamma) / (1 - gamma ** N)

N = 10
r_values = [1.5, 2.0, 3.0]
w_range = np.linspace(0, 1, 200)

plt.figure(figsize=(6, 4))
for r in r_values:
    plt.plot(w_range, [rho_1(w, r, N) for w in w_range], label=f"$r = {r}$")
plt.axhline(1 / N, color="gray", linestyle="--", label=f"Neutral drift $1/N =
1/{N}$")
plt.xlabel("Selection intensity $w$")
plt.ylabel(r"Fixation probability $\rho_1$")
plt.title(f"Fixation probability vs selection intensity ($N={N}$)")
plt.legend()
plt.tight_layout()
```



Solution 11.5: Solution to Exercise 5

1. The standard definition of the Prisoners Dilemma is:

$$M_r = \begin{pmatrix} R & S \\ T & P \end{pmatrix} \tag{11.48}$$

with $T > R > P > S$ and $2R > T + S$. In this case we have:

$$R = 3a \qquad T = 2a \qquad S = a \qquad P = 2 \tag{11.49}$$

This implies that for $a > 0$ we have $T < R$ thus this is not Prisoner's Dilemma.

2. There are 5 potential Nash equilibria for

$$M_r = \begin{pmatrix} 3a & a \\ 2a & 2 \end{pmatrix} \qquad M_c = \begin{pmatrix} 3a & 2a \\ a & 2 \end{pmatrix} \tag{11.50}$$

(r_1, c_1) is a Nash Equilibrium if and only if:

$$3a \geq 2a \Leftrightarrow a \geq 0 \tag{11.51}$$

(r_1, c_2) is a Nash Equilibrium if and only if:

$$a \geq 2 \quad \text{and} \quad 2a \geq 3a \Rightarrow a \geq 0 \tag{11.52}$$

This is not possible.

(r_2, c_1) is a Nash equilibrium if and only if:

$$2a \geq 3a \quad \text{and} \quad a \geq 2a \tag{11.53}$$

Which is again, not possible.

(r_2, c_2) is a Nash equilibrium if and only if:

$$a \leq 2 \tag{11.54}$$

Let $\sigma_1 = (x, 1-x)$ and $\sigma_2 = (y, 1-y)$.

(σ_1, σ_2) with $0 < x < 1$ and $0 < y < 1$ is a Nash equilibrium if and only if:

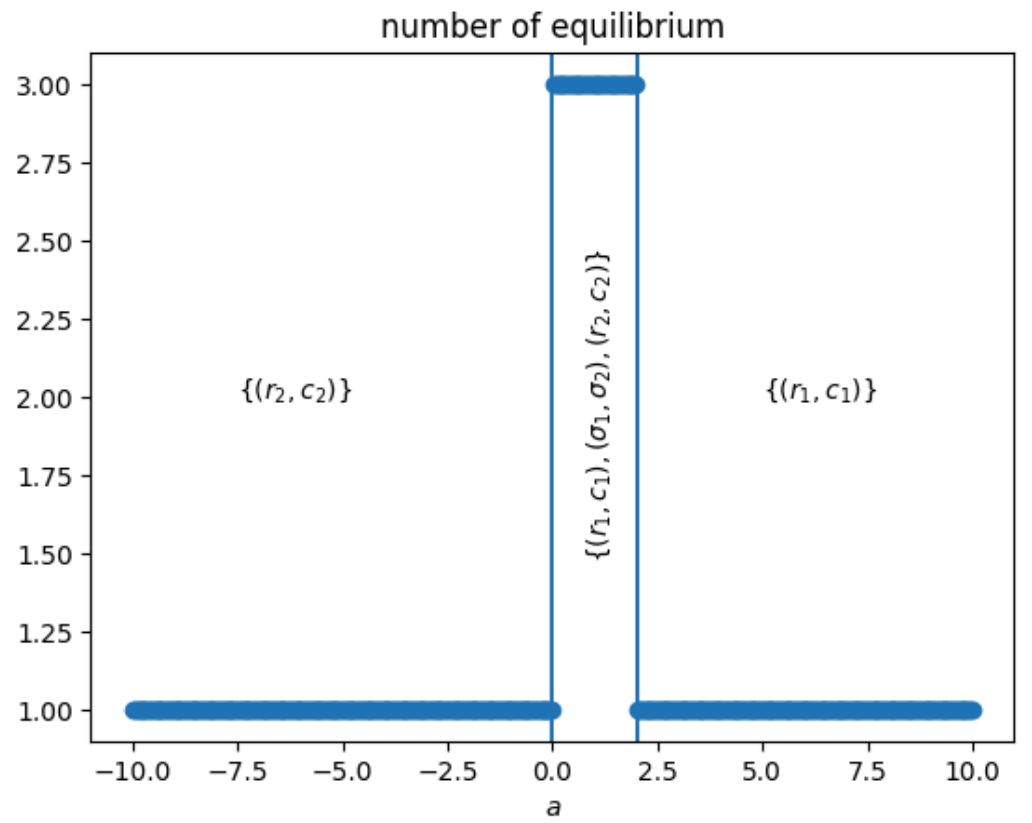
$$\begin{aligned} 3ay + a(1-y) &= 2ay + 2(1-y) \\ 2ay + a &= 2ay + 2 - 2y \\ y(2a - 2a + 2) &= 2 - a \\ y &= \frac{2-a}{2} \end{aligned} \tag{11.55}$$

$0 < y < 1$ holds if and only if:

$$0 < a < 2 \tag{11.56}$$

The above answers the question. Here is some code to count and display the final situation:

```
import nashpy as nash
import matplotlib.pyplot as plt
import numpy as np

a_values = np.linspace(-10, 10, 500)
number_of_equilibrium = []
for a in a_values:
    M_r = np.array(
        (
            (3 * a, a),
            (2 * a, 2),
        )
    )
    M_c = M_r.T
    game = nash.Game(M_r, M_c)
    number_of_equilibrium.append(
        len(
            list(game.support_enumeration())
        )
    )

plt.figure()
plt.scatter(a_values, number_of_equilibrium)
plt.xlabel("$a$")
plt.title("number of equilibrium")
plt.axvline(2)
plt.axvline(0)
plt.text(-7.5, 2, r"$\{(r_2, c_2)\}$")
plt.text(5, 2, r"$\{(r_1, c_1)\}$")
plt.text(0.75, 1.5, r"$\{(r_1, c_1), (\sigma_1, \sigma_2),(r_2, c_2) \}$",
rotation=90);
```



4. For $N = 2$:

$$
\begin{aligned}
f_1(a) &= \frac{(i-1)3a + (2-i)a}{1} \\
f_2(a) &= \frac{2ai + 2(1-i)}{1}
\end{aligned}
\tag{11.57}
$$

This gives:

$$
\gamma_i = \frac{i(2a-2)+2}{2ai-a} = \frac{2ai-2i+2}{2ai-a} \tag{11.58}
$$

This gives:

$$
\rho_1 = \frac{1}{1+\gamma_1} = \frac{1}{1+\frac{2a}{a}} = \frac{1}{1+2} = \frac{1}{3} \tag{11.59}
$$

Let us use some code to confirm these calculations:

```
import sympy as sym

i = sym.Symbol("i")
a = sym.Symbol("a")
f_1 = ((3 * a * (i - 1) + (2 - i) * a) / 2)
f_2 = ((2 * a * i + 2 * (1 - i)) / 2)
gamma = sym.simplify(f_2 / f_1)
gamma
```

```
2*(-a*i + i - 1)/(a*(1 - 2*i))
```

```
rho_1_N_2 = 1 / (1 + gamma.subs({i: 1}))
rho_1_N_2
```

```
1/3
```

For $N = 3$:

$$\begin{aligned} f_1(a) &= \frac{(i-1)3a + (3-i)a}{2} \\ f_2(a) &= \frac{2ai + 2(2-i)}{2} \end{aligned} \tag{11.60}$$

$$\gamma_i = \frac{2ai + 4 - 2i}{3ai - 3a + 3a - ai} = \frac{2ai + 4 - 2i}{2ai} = \frac{i(a-1) + 2}{ai} \tag{11.61}$$

This gives:

$$\begin{aligned} \rho_1 &= \frac{1}{1 + \gamma_1 + \gamma_1\gamma_2} = \frac{1}{1 + \frac{a+1}{a} + \frac{a+1}{a}\frac{2a}{2a}} = \frac{1}{\frac{a+2a+2}{a}} \\ &= \frac{a}{3a+2} \end{aligned} \tag{11.62}$$

Let us use some code to confirm these calculations:

```
f_1 = ((3 * a * (i - 1) + (3 - i) * a) / 2)
f_2 = ((2 * a * i + 2 * (2 - i)) / 2)
gamma = sym.simplify(f_2 / f_1)
gamma
```

```
(a*i - i + 2)/(a*i)
```

```
rho_1_N_3 = 1 / (1 + gamma.subs({i: 1})+ gamma.subs({i: 1}) * gamma.subs({i:
2}))
sym.simplify(rho_1_N_3)
```

```
a/(3*a + 2)
```

For $N = 4$:

$$\begin{aligned} f_1(a) &= \frac{(i-1)3a + (4-i)a}{3} \\ f_2(a) &= \frac{2ai + 2(3-i)}{3} \end{aligned} \tag{11.63}$$

$$\gamma_i = \frac{i(2a-2) + 6}{2ai + a} = \frac{2(ai - i + 3)}{a(2i+1)} \tag{11.64}$$

This gives:

$$
\begin{aligned}
\rho_1 &= \frac{1}{1+\gamma_1+\gamma_1\gamma_2+\gamma_1\gamma_2\gamma_3} \\
&= \frac{1}{1+\frac{2(a+2)}{3a}+\frac{2(a+2)}{3a}\frac{2(2a+1)}{5a}+\frac{2(a+2)}{3a}\frac{2(2a+1)}{5a}\frac{6a}{7a}} \\
&= \frac{1}{\frac{3\cdot5a^2}{3\cdot5a^2}+\frac{2(a+2)\cdot5a}{3\cdot5a^2}+\frac{\frac{7+6}{7}(2(a+2)2(2a+1))}{3\cdot5a^2}} \\
&= \frac{1}{\frac{3\cdot5\cdot7a^2}{3\cdot5\cdot7a^2}+\frac{7\cdot2(a+2)\cdot5a}{3\cdot5\cdot7a^2}+\frac{13(2(a+2)2(2a+1))}{3\cdot5\cdot7a^2}} \\
&= \frac{3\cdot5\cdot7a^2}{3\cdot5\cdot7a^2+7\cdot2(a+2)\cdot5a+13(2(a+2)2(2a+1))} \\
&= \frac{3\cdot5\cdot7a^2}{105a^2+70a^2+140a+104a^2+260a+104} \\
&= \frac{3\cdot5\cdot7a^2}{279a^2+400a+104}
\end{aligned}
\tag{11.65}
$$

Let us use some code to confirm these calculations:

```
f_1 = ((3 * a * (i - 1) + (4 - i) * a) / 3)
f_2 = ((2 * a * i + 2 * (3 - i)) / 3)
gamma = sym.simplify(f_2 / f_1)
gamma
```

```
2*(a*i - i + 3)/(a*(2*i + 1))
```

```
rho_1_N_4 = 1 / (1 + gamma.subs({i: 1})+ gamma.subs({i: 1}) * gamma.subs({i:
2})+ gamma.subs({i: 1}) * gamma.subs({i: 2}) * gamma.subs({i: 3}))
sym.simplify(rho_1_N_4)
```

```
105*a**2/(279*a**2 + 400*a + 104)
```

5.

a. For all $N = 2$ we have:

$$\lim_{a\to 0}\rho_1 = 1/3 \tag{11.66}$$

For $N \in \{3, 4\}$

$$\lim_{a\to 0}\rho_1 = 0 \tag{11.67}$$

b. For $N = 2$ we have:

$$\lim_{a\to\infty}\rho_1 = \lim_{a\to\infty} 1/3 = 1/3 \tag{11.68}$$

For $N = 3$ we have:

$$\lim_{a\to\infty}\rho_1 = \lim_{a\to\infty}\frac{a}{3a+2} = \lim_{a\to\infty}\frac{1}{3+2/a} = 1/3 \tag{11.69}$$

For $N = 4$ we have:

$$\begin{aligned}\lim_{a\to\infty}\rho_1 &= \lim_{a\to\infty}\frac{3\cdot 5\cdot 7a^2}{279a^2+400a+104}\\ &= \lim_{a\to\infty}\frac{3\cdot 5\cdot 7}{279+400/a+104/a^2}\\ &= \frac{3\cdot 5\cdot 7}{279} = \frac{3\cdot 5\cdot 7}{3^2\cdot 31} = \frac{35}{93}\end{aligned} \tag{11.70}$$

Let us write some code to confirm this:

```
sym.limit(rho_1_N_2, a, 0)
```

```
1/3
```

```
sym.limit(rho_1_N_2, a, sym.oo)
```

```
1/3
```

```
sym.limit(rho_1_N_3, a, 0)
```

```
0
```

```
sym.limit(rho_1_N_3, a, sym.oo)
```

```
1/3
```

```
sym.limit(rho_1_N_4, a, 0)
```

```
0
```

```
sym.limit(rho_1_N_4, a, sym.oo)
```

```
35/93
```

This shows that the amplification of the fitness (the increase of a) has a greater effect for larger N.

12. Learning and Evolutionary Dynamics

The replicator dynamics and Moran process are just two of many ways a population might update its strategies over time. This chapter examines a broader family of learning and evolutionary dynamics, exploring how the choice of update rule shapes long-run behaviour.

Figure 12.1: Ideas spread through a population as individuals copy what appears to be working. A number of population update mechanisms exist that model this spread in different ways.

12.1 Motivating Example: Does the Update Rule Matter?

Return to the graduate student reading group from Motivating Example: Everyone's citing the preprint. The same coordination game governs citations, textbook (T) versus preprint (P), but now consider three plausible ways students might update their habits:

1. **Imitation**: a student notices a peer doing well and copies them.
2. **Best-response**: a student calculates which citation would be optimal given current group behaviour and switches if warranted.
3. **Generational turnover**: new students join each cohort and tend to adopt whichever citation style is more prevalent.

The payoff matrix is the same in all three cases. Yet these mechanisms can produce different trajectories, different fixation probabilities, and different long-run distributions. This raises a fundamental question: **does the update rule matter?**

The answer is yes, particularly in small populations, at high selection intensity, or when the game has multiple equilibria. This chapter introduces three families of update rules (imitation, introspection, and best-response) alongside the Wright–Fisher process, and shows how they relate to each other and to the replicator dynamics and Moran process studied in previous chapters.

12.2 Theory

12.2.1 Definition: Imitation Dynamics

In **imitation dynamics** a population of N individuals updates strategy by pairwise comparison. At each step:

1. Two individuals i and j are selected uniformly at random.
2. Individual i switches to j's strategy with the **Fermi probability**:

$$P(i \to j) = \frac{1}{1 + e^{-\beta(\pi_j - \pi_i)}} \tag{12.1}$$

where π_i and π_j are their current payoffs and $\beta \geq 0$ is the **selection intensity**.

The parameter β controls how strongly payoff differences drive imitation:

- $\beta = 0$: strategies are copied uniformly at random (neutral drift).
- $\beta \to \infty$: copying is deterministic; the higher-payoff strategy is always adopted.

Note

The Fermi function $1/(1+e^{-x})$ is borrowed from statistical physics, where it describes the probability of a particle occupying a higher-energy state. Here it models the "temperature" of decision-making: high β means cold (rational), low β means hot (noisy).

12.2.1.1 Example: Imitation in the Citation Game

For the citation coordination game with $\beta = 1$, suppose the group has $N = 4$ students with 2 citing the textbook and 2 citing the preprint. Each student's fitness is:

$$\pi_T = \frac{1 \cdot 3 + 2 \cdot 0}{3} = 1 \qquad \pi_P = \frac{2 \cdot 1 + 1 \cdot 2}{3} = \frac{4}{3} \tag{12.2}$$

If student i cites T and is paired with student j citing P, the probability that i switches to P is:

$$\begin{aligned} P(i \to j) &= \frac{1}{1 + e^{-1 \cdot (4/3 - 1)}} \\ &= \frac{1}{1 + e^{-1/3}} \approx 0.582 \end{aligned} \tag{12.3}$$

Compare this to the Moran process, where the probability of the preprint spreading would instead depend on relative fitness proportions. Both models favour P in this state, but at different rates and with different stochastic properties.

12.2.2 Definition: Introspection Dynamics

In **introspection dynamics** individuals update by counterfactual reasoning rather than by observing others. At each step:

1. One individual i is selected uniformly at random.
2. Individual i computes the payoff π_j they *would* receive if they switched to each alternative strategy j, holding all other individuals' strategies fixed.
3. They switch to strategy j with the Fermi probability:

$$P(i \to j) = \frac{1}{1 + e^{-\beta(\pi_j - \pi_i)}} \tag{12.4}$$

The key distinction from imitation is the information used: imitation requires observing another individual's payoff; introspection requires only computing a hypothetical payoff from the current population composition.

12.2.2.1 Example: Introspection vs Imitation

In the citation game at state $(v_T, v_P) = (2, 2)$, a textbook-citing student using introspection computes:

- Current payoff: $\pi_T = 1$ (playing T against 1 T-player and 2 P-players among the remaining 3 partners).
- Hypothetical payoff if switching: $\pi_P = (2 \cdot 1 + 1 \cdot 2)/3 = 4/3$ (playing P against the 2 P-players and 1 T-player).

The switching probability is the same as in the imitation example above, ≈ 0.582. In symmetric games with homogeneous populations the two dynamics coincide; differences emerge in asymmetric games or structured populations where who you observe versus who you *are* matters.

12.2.3 Definition: Best-Response Dynamics

Best-response dynamics is the deterministic limit of introspection dynamics as $\beta \to \infty$. At each step an individual switches to whichever strategy has the highest expected payoff against the current population:

$$P(i \to j) = \tag{12.5}$$

In the infinite-population continuous-time limit, the population share x_i of strategy i evolves as (Gilboa & Matsui, 1991):

$$\dot{x}_i = \mathbf{1}\left[\pi_i = \max_k \pi_k\right] - x_i \tag{12.6}$$

Best-response dynamics is used in economic models where agents are fully rational and have perfect information about population composition. It always converges to a Nash equilibrium or cycles around one, but unlike replicator dynamics it can reach strict equilibria from any initial condition.

12.2.4 Definition: Wright–Fisher Process

The **Wright–Fisher process** models evolution in a population with non-overlapping generations of fixed size N. Each generation is formed by sampling N offspring independently from the previous generation, where strategy i is chosen with probability proportional to its fitness:

$$p_i = \frac{f_i}{\sum_k f_k} \tag{12.7}$$

Optionally, each offspring mutates to a uniformly chosen strategy with probability μ.

Unlike the Moran process, which replaces one individual at a time, the Wright–Fisher process replaces the entire population simultaneously. This makes it the natural model for organisms with discrete, synchronised generations (annual plants, insects, some microbes). The biological origins of this process, and of the Moran process, are discussed in The Biological Origins of Evolutionary Game Theory.

12.2.4.1 Example: One Generation of the Wright–Fisher Process

For the citation game at state $(v_T, v_P) = (3, 1)$ with $N = 4$ and $\beta = 1$ (using exponential fitness $f_i = e^{\beta \pi_i}$):

$$f_T = e^{6/3} = e^2 \approx 7.39 \qquad f_P = e^{3/3} = e^1 \approx 2.72 \tag{12.8}$$

The probability that a randomly sampled offspring cites T is:

$$p_T = \frac{7.39}{7.39 + 2.72} \approx 0.731 \tag{12.9}$$

The next generation (v_T', v_P') follows a Binomial$(4,\ 0.731)$ distribution, so for example $P(v_T' = 4) \approx 0.731^4 \approx 0.286$.

12.2.5 Theorem: Mean-Field Limit

For all four dynamics (imitation, introspection, best-response, and Wright–Fisher) in the limit of large population size $N \to \infty$ and weak selection $\beta \to 0$ (with βN held fixed), the expected change in strategy frequencies converges to the **replicator equation** (Hofbauer & Sigmund, 1998; Traulsen et al., 2005):

$$\dot{x}_i = x_i(\pi_i - \bar{\pi}), \qquad \bar{\pi} = \sum_k x_k \pi_k \tag{12.10}$$

This result explains why replicator dynamics occupies a central place in evolutionary game theory: it is the universal mean-field limit shared by all these distinct microscopic update rules. Differences between dynamics matter most in small populations or at high selection intensity.

12.3 Exercises

Exercise 12.1:

Consider the Prisoner's Dilemma with $R = 3, S = 0, T = 5, P = 1$ and a population of $N = 4$ with 2 cooperators and 2 defectors. Each individual interacts with all $N - 1$ others.

1. Compute the average payoff π_C and π_D at this state.
2. Using $\beta = 1$, compute the Fermi probability that a cooperator imitates a defector, and vice versa.
3. What happens to these probabilities as $\beta \to 0$ and $\beta \to \infty$?

Exercise 12.2:

Consider the coordination game $M = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ with $N = 4$ individuals, starting from state $(v_1, v_2) = (2, 2)$.

1. Under imitation dynamics with $\beta = 2$, compute the probability that the number of type-1 individuals increases by 1.
2. Under the Moran process, compute the same probability.
3. Under the Wright–Fisher process (one generation, using $f_i = e^{\beta \pi_i}$ with $\beta = 2$), what is the probability of the next state being $(3, 1)$?
4. Which dynamic most strongly favours strategy 1 in this state?

Exercise 12.3:

Consider the coordination game with payoff matrix

$$M = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \tag{12.11}$$

in the continuous best-response dynamics with population shares $x_1, x_2 = 1 - x_1$.

1. Find all rest points of the dynamics.
2. Determine the stability of each rest point.
3. Compare to the Nash equilibria of the game.

Exercise 12.4:

Consider the Stag Hunt game:

$$M = \begin{pmatrix} 4 & 0 \\ 1 & 2 \end{pmatrix} \tag{12.12}$$

Both $(1, 0)$ and $(0, 1)$ are strict Nash equilibria (risk-dominant: $(0, 1)$; payoff-dominant: $(1, 0)$).

1. Under replicator dynamics, which equilibrium does the system converge to from the initial condition $x_1 = 0.4$?
2. Under best-response dynamics, what is the basin of attraction of each equilibrium?
3. Briefly explain why best-response dynamics and replicator dynamics can select different equilibria in games with multiple strict Nash equilibria.

12.4 Programming

The `ludics` library represents an evolutionary game as a finite-population Markov chain and computes fixation probabilities and stationary distributions exactly. It implements the Moran process, Fermi imitation, and introspection dynamics directly, and its transition-matrix builder accepts a custom update rule, which lets us add the Wright–Fisher process as well.

12.4.1 Setting up the citation game

We work with the citation coordination game on a population of $N = 4$ students. A state records each individual's strategy (0 for textbook, 1 for preprint), and the fitness of an individual is its average payoff against the rest of the population.

```
import numpy as np
import ludics

payoff_matrix = np.array([[3, 0], [1, 2]])  # citation game: T = 0, P = 1


def citation_fitness(state, payoff_matrix, **kwargs):
    """Average payoff of each individual against all others."""
    number_of_individuals = len(state)
    return np.array([
        np.mean([
            payoff_matrix[state[individual], state[other]]
            for other in range(number_of_individuals)
            if other != individual
        ])
        for individual in range(number_of_individuals)
    ])


state_space = ludics.get_state_space(N=4, k=2)
```

```
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:8:
SyntaxWarning: "\i" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\i"? A raw string is also an option.
  We have a state v \in S = [$v_1$, .... $v_n$] as the set of types of
individuals in the population, so that $v_i$ is the type of individual i. |S| =
$k^N$
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:52:
SyntaxWarning: "\s" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\s"? A raw string is also an option.
  $\frac{\sum_{v_i = u_{i*}}{f(v_i)}}{\sum_{v_i}f(v_i)}$
```

```
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:85:
SyntaxWarning: "\p" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\p"? A raw string is also an option.
  returns $\phi(a_i, a_j) = \frac{1}{1 + \exp({\frac{f(a_{i}) - f(a_{j})
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:121:
SyntaxWarning: "\s" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\s"? A raw string is also an option.
  $\sum_{a_j=b_{i^*}}^N\frac{1}{N(N-1)}\phi(f_i(a) - f(a_j))$
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:177:
SyntaxWarning: "\s" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\s"? A raw string is also an option.
  $\frac{1}{N}\frac{\sum_{a_{j} = b_{I(\textbf{a}, \textbf{b})}}f_j(\textbf{a})}
{\sum_{k}f_k(\textbf{a})}\phi(\Delta(f_{I(\textbf{a,b})}))$
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:234:
SyntaxWarning: "\p" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\p"? A raw string is also an option.
  $\frac{1}{N(m_j - 1)}\phi(f_i(a) - f_i(b))$
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:300:
SyntaxWarning: "\p" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\p"? A raw string is also an option.
  will choose the higher fitness strategy in $\phi$
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/ludics/main.py:784:
SyntaxWarning: "\m" is an invalid escape sequence. Such sequences will not work
in the future. Did you mean "\\m"? A raw string is also an option.
  strategies) array where $\mu_{ik}$ gives the probability of player $i$
```

12.4.2 Building each dynamic as a Markov chain

Moran, Fermi imitation, and introspection are built in. Wright–Fisher is not, but because each of the N offspring of a generation is drawn independently in proportion to parental fitness, its transition probability factorises as a product over offspring, and `generate_transition_matrix` accepts it as a custom rule.

```python
def wright_fisher_transition_probability(
    source, target, fitness_function, selection_intensity, **kwargs
):
    """Wright–Fisher: each offspring is sampled independently, in proportion
    to parental fitness (non-overlapping generations)."""
    fitness = 1 + selection_intensity * fitness_function(source, **kwargs)
    total_fitness = fitness.sum()
    probability = 1.0
    for offspring_type in target:
        probability *= fitness[source == offspring_type].sum() / total_fitness
    return probability


moran = ludics.generate_transition_matrix(
    state_space, citation_fitness, ludics.compute_moran_transition_probability,
    selection_intensity=1.0, payoff_matrix=payoff_matrix,
)
fermi = ludics.generate_transition_matrix(
    state_space, citation_fitness, ludics.compute_fermi_transition_probability,
    choice_intensity=1.0, payoff_matrix=payoff_matrix,
)
wright_fisher = ludics.generate_transition_matrix(
```

```
    state_space, citation_fitness, wright_fisher_transition_probability,
    selection_intensity=1.0, payoff_matrix=payoff_matrix,
)
introspection = ludics.generate_transition_matrix(
    state_space, citation_fitness,
    ludics.compute_introspection_transition_probability,
    choice_intensity=1.0, number_of_strategies=2, payoff_matrix=payoff_matrix,
)
```

12.4.3 Does the update rule matter?

The three absorbing dynamics, Moran, Fermi imitation, and Wright–Fisher, each end in fixation: every individual ends up citing either the textbook or the preprint. We read off the probability that a single preprint adopter eventually takes over, and it differs from one rule to the next, which is the central message of this chapter.

```
single_adopter = next(i for i, s in enumerate(state_space) if s.sum() == 1)
all_preprint = next(i for i, s in enumerate(state_space) if s.sum() == 4)

for name, matrix in [("Moran", moran), ("Fermi imitation", fermi),
                     ("Wright-Fisher", wright_fisher)]:
    absorption = ludics.get_absorption_probabilities(matrix, state_space)
    pairs = np.array(absorption[single_adopter]).reshape(-1, 2)
    fixation = next(p for index, p in pairs if int(index) == all_preprint)
    print(f"{name}: fixation probability of one preprint adopter =
{fixation:.4f}")
```

```
Moran: fixation probability of one preprint adopter = 0.2342
Fermi imitation: fixation probability of one preprint adopter = 0.1656
Wright-Fisher: fixation probability of one preprint adopter = 0.1729
```

Introspection dynamics, by contrast, is ergodic: it never fixes but visits every state indefinitely, so the natural summary is its stationary distribution.

```
stationary = ludics.compute_steady_state(introspection)
print(
    "Introspection: stationary probability of all-preprint =",
    round(float(stationary[all_preprint]), 4),
)
```

```
Introspection: stationary probability of all-preprint = 0.2857
```

Note

The `ludics` cells above build the entire Markov chain and extract exact fixation probabilities and stationary distributions, whereas the exercises and their solutions work directly with the per-step transition probabilities by hand. The two approaches describe the same dynamics but adopt different conventions, so their numbers need not coincide exactly. In particular, `ludics` parametrises the Fermi and introspection rules by a `choice_intensity` (an inverse temperature, roughly $1/\beta$) and the Moran process by a fitness of the form $1 + \delta\,\pi$, while the worked examples use β directly, proportional fitness $f = \pi$ for the Moran process, and exponential fitness $f = e^{\beta\pi}$ for the Wright–Fisher process. Each is a standard choice, and the qualitative comparison between update rules is unaffected.

12.5 Notable Research

The Fermi imitation rule used in this chapter was proposed and analysed by (Traulsen et al., 2006), who showed that pairwise comparison dynamics reduces to the replicator equation in the large-population, weak-selection limit. This paper is foundational to the study of evolutionary dynamics in finite populations under social learning.

Introspection dynamics is a more recent addition to this family, introduced by (Couto et al., 2022). Unlike imitation, introspection does not require observing a specific partner's outcome, only reasoning about what *would* happen under an alternative strategy. This distinction becomes particularly important in asymmetric games and has applications to human behavioural experiments where direct comparison is not always possible.

Best-response dynamics has roots in the economic learning literature, tracing back to (Gilboa & Matsui, 1991). Its relationship to replicator dynamics and its convergence properties in potential games are thoroughly analysed in (Hofbauer & Sigmund, 1998).

The Wright–Fisher process predates the Moran process in the biological literature, originating in the foundational work of Fisher (Fisher, 1930) and Wright (Wright, 1931) on genetic drift. The key result that both processes converge to replicator dynamics in the large-population limit was established in (Traulsen et al., 2005).

12.6 Conclusion

This chapter surveyed four update rules (imitation, introspection, best-response, and Wright–Fisher) as alternatives to the Moran process and replicator dynamics covered in preceding chapters. Together they form a landscape of evolutionary and learning models that share a common mean-field limit (the replicator equation) but differ in their finite-population and high-selection behaviour.

The choice of update rule is not merely a mathematical convention. It reflects an assumption about how real agents (biological organisms, humans, firms) actually change their behaviour: by copying successful peers, by rational introspection, by best-responding to the current environment, or through generational turnover. Understanding when these assumptions differ in their predictions is essential for applying evolutionary game theory to empirical settings.

Table 12.1 summarises the key concepts.

Dynamic	Update mechanism	Finite population?	Mean-field limit
Replicator	Continuous frequency change proportional to relative payoff	No (infinite)	n/a (is the limit)
Moran	Birth proportional to fitness; death uniform	Yes	Replicator
Imitation	Fermi pairwise copying with selection intensity β	Yes	Replicator
Introspection	Counterfactual switching via Fermi rule	Yes	Replicator
Best-response	Deterministic switch to highest-payoff strategy	No (or $\beta \to \infty$)	Replicator
Wright–Fisher	Multinomial sampling proportional to fitness per generation	Yes	Replicator

Table 12.1: Summary of learning and evolutionary dynamics

Attention

All of the dynamics in this chapter, and in the two preceding chapters, converge to the replicator equation as the population becomes large and selection becomes weak. The replicator equation is therefore not just one model among many: it is the universal mean-field description of evolutionary and social learning. The distinctive predictions of finite-population models emerge precisely where this approximation breaks down.

12.7 Solutions

Solution 12.1: Solution to Exercise 1

1. With $N = 4$, 2 cooperators (C) and 2 defectors (D), each individual interacts with all $N - 1 = 3$ others.

For a cooperator, the opponents are 1 cooperator and 2 defectors:

$$\pi_C = \frac{1 \cdot R + 2 \cdot S}{3} = \frac{1 \cdot 3 + 2 \cdot 0}{3} = 1 \tag{12.13}$$

For a defector, the opponents are 2 cooperators and 1 defector:

$$\pi_D = \frac{2 \cdot T + 1 \cdot P}{3} = \frac{2 \cdot 5 + 1 \cdot 1}{3} = \frac{11}{3} \tag{12.14}$$

1. Using the Fermi probability with $\beta = 1$:

The probability that a cooperator imitates a defector (switching from C to D):

$$\begin{aligned} P(C \to D) &= \frac{1}{1 + e^{-\beta(\pi_D - \pi_C)}} \\ &= \frac{1}{1 + e^{-(11/3 - 1)}} \\ &= \frac{1}{1 + e^{-8/3}} \approx \frac{1}{1 + 0.0695} \approx 0.935 \end{aligned} \tag{12.15}$$

The probability that a defector imitates a cooperator (switching from D to C):

$$\begin{aligned} P(D \to C) &= \frac{1}{1 + e^{-\beta(\pi_C - \pi_D)}} \\ &= \frac{1}{1 + e^{-(1 - 11/3)}} \\ &= \frac{1}{1 + e^{8/3}} \approx \frac{1}{1 + 14.39} \approx 0.065 \end{aligned} \tag{12.16}$$

Note that $P(C \to D) + P(D \to C) = 1$, as expected from the symmetry $P(i \to j) = 1 - P(j \to i)$.

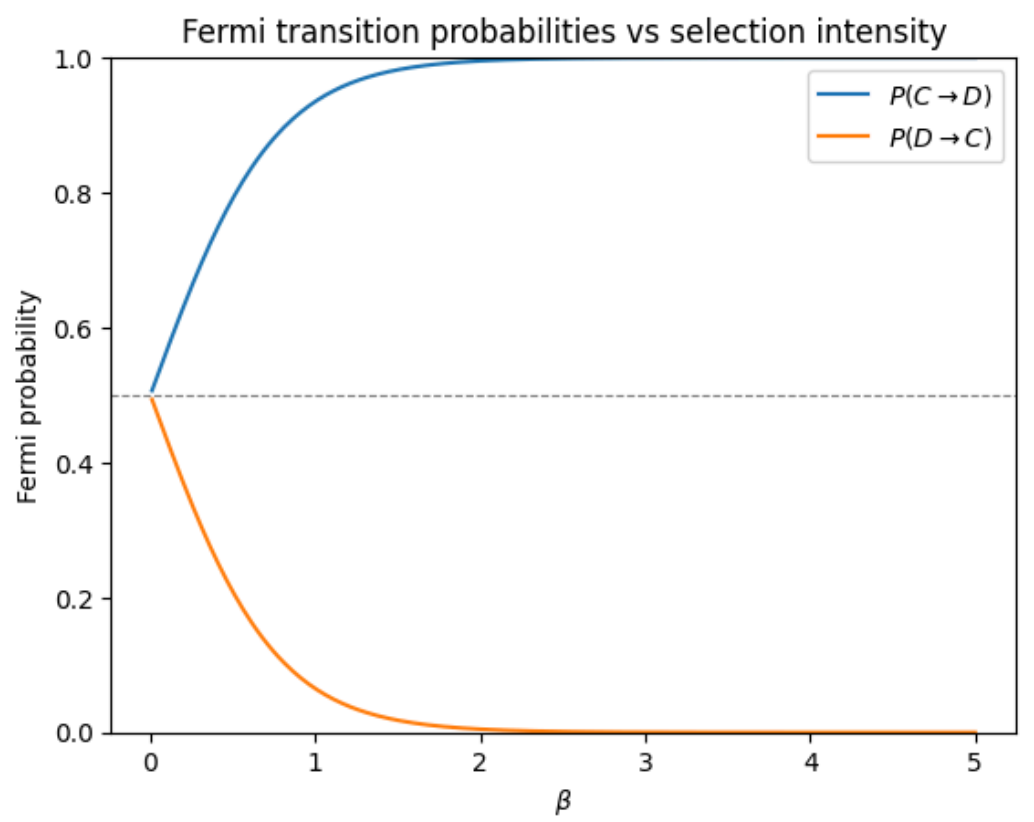
1. As $\beta \to 0$: Both Fermi probabilities converge to $1/2$. The exponential term $e^{-\beta \Delta\pi} \to 1$ regardless of the payoff difference $\Delta\pi$, so $P(i \to j) \to 1/(1+1) = 1/2$. Imitation becomes purely random, neutral drift.

As $\beta \to \infty$: The higher-payoff individual is always imitated. Since $\pi_D > \pi_C$, we get $P(C \to D) \to 1$ and $P(D \to C) \to 0$. Defectors are always imitated and cooperators never are; the dynamics are deterministic and defection takes over.

```python
import numpy as np

R, S, T, P_pd = 3, 0, 5, 1
N = 4
n_C, n_D = 2, 2

pi_C = (n_C - 1) * R + n_D * S
pi_C /= (N - 1)
pi_D = n_C * T + (n_D - 1) * P_pd
pi_D /= (N - 1)

print(f"pi_C = {pi_C}")
print(f"pi_D = {pi_D:.4f}")

beta = 1
P_C_to_D = 1 / (1 + np.exp(-beta * (pi_D - pi_C)))
P_D_to_C = 1 / (1 + np.exp(-beta * (pi_C - pi_D)))
print(f"P(C->D) = {P_C_to_D:.4f}")
print(f"P(D->C) = {P_D_to_C:.4f}")
```

```
pi_C = 1.0
pi_D = 3.6667
P(C->D) = 0.9350
P(D->C) = 0.0650
```

```python
import matplotlib.pyplot as plt

beta_values = np.linspace(0.01, 5, 300)
P_C_to_D_vals = 1 / (1 + np.exp(-beta_values * (pi_D - pi_C)))
P_D_to_C_vals = 1 / (1 + np.exp(-beta_values * (pi_C - pi_D)))

plt.figure()
plt.plot(beta_values, P_C_to_D_vals, label=r"$P(C \to D)$")
plt.plot(beta_values, P_D_to_C_vals, label=r"$P(D \to C)$")
plt.axhline(0.5, color="gray", linestyle="--", linewidth=0.8)
plt.xlabel(r"$\beta$")
plt.ylabel("Fermi probability")
plt.title("Fermi transition probabilities vs selection intensity")
plt.legend()
plt.ylim(0, 1);
```



Solution 12.2: Solution to Exercise 2

The game is $M = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ with $N = 4$ and state $(v_1, v_2) = (2, 2)$.

Each individual interacts with all $N - 1 = 3$ others.

Payoffs at state $(2, 2)$:

For a type-1 individual (1 ally among 3 opponents):

$$\pi_1 = \frac{1 \cdot 2 + 2 \cdot 0}{3} = \frac{2}{3} \tag{12.17}$$

For a type-2 individual (2 opponents of type 1, 1 ally):

$$\pi_2 = \frac{2 \cdot 0 + 1 \cdot 1}{3} = \frac{1}{3} \tag{12.18}$$

1. **Imitation dynamics with $\beta = 2$.**

For the number of type-1 individuals to increase by 1, a type-2 individual must be selected as i, a type-1 individual must be selected as j, and i must adopt j's strategy.

Probability of selecting a type-2 individual as i and a type-1 individual as j:

$$\frac{v_2}{N} \cdot \frac{v_1}{N-1} = \frac{2}{4} \cdot \frac{2}{3} = \frac{1}{3} \tag{12.19}$$

Fermi probability that the type-2 individual imitates the type-1 individual:

$$\begin{aligned} P(2 \to 1) &= \frac{1}{1+e^{-\beta(\pi_1-\pi_2)}} \\ &= \frac{1}{1+e^{-2(2/3-1/3)}} \\ &= \frac{1}{1+e^{-2/3}} \approx 0.661 \end{aligned} \tag{12.20}$$

Therefore the probability of increasing type-1 count by 1:

$$P(\text{increase}) = \frac{1}{3} \times \frac{1}{1+e^{-2/3}} \approx 0.220 \tag{12.21}$$

1. **Moran process.**

In the Moran process, fitness is proportional to payoff (using linear fitness $f_i = 1 + \delta\pi_i$ or simply $f_i = \pi_i$ with normalisation). Using proportional fitness $f_i = \pi_i$:

$$p_{2\to1} = \frac{v_1 \cdot \pi_1}{v_1\pi_1 + v_2\pi_2} \cdot \frac{v_2}{N-1} \tag{12.22}$$

The probability that a type-1 is selected for reproduction and a type-2 for removal:

$$\begin{aligned} P(\text{increase}) &= \frac{v_1\pi_1}{v_1\pi_1 + v_2\pi_2} \cdot \frac{v_2}{N} \\ &= \frac{2 \cdot (2/3)}{2 \cdot (2/3) + 2 \cdot (1/3)} \cdot \frac{2}{4} \\ &= \frac{4/3}{4/3 + 2/3} \cdot \frac{1}{2} \\ &= \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3} \end{aligned} \tag{12.23}$$

1. **Wright-Fisher process (one generation, $f_i = e^{\beta\pi_i}$ with $\beta = 2$).**

Exponential fitnesses:

$$f_1 = e^{2\cdot 2/3} = e^{4/3} \approx 3.794 \qquad f_2 = e^{2\cdot 1/3} = e^{2/3} \approx 1.948 \tag{12.24}$$

Sampling probability for type-1:

$$\begin{aligned} p_1 &= \frac{v_1 f_1}{v_1 f_1 + v_2 f_2} \\ &= \frac{2e^{4/3}}{2e^{4/3} + 2e^{2/3}} \\ &= \frac{e^{4/3}}{e^{4/3} + e^{2/3}} \\ &= \frac{1}{1+e^{-2/3}} \approx 0.661 \end{aligned} \tag{12.25}$$

The next generation (v_1', v_2') follows Binomial$(4, p_1)$. The probability of the next state being $(3, 1)$ is:

$$\begin{aligned} P(v_1' = 3) &= \texttt{binom}\ 4 3 p_1^3 (1-p_1)^1 \\ &= 4 \times (0.661)^3 \times (0.339) \approx 4 \times 0.289 \times 0.339 \approx 0.392 \end{aligned} \tag{12.26}$$

1. **Comparison.**

- Imitation dynamics: $P(\text{increase}) \approx 0.220$
- Moran process: $P(\text{increase}) = 1/3 \approx 0.333$
- Wright-Fisher (probability of state $(3,1)$): ≈ 0.392

The Wright-Fisher process most strongly favours strategy 1 in this state, followed by the Moran process, then imitation dynamics. The difference arises because the Wright-Fisher process uses exponential fitness (amplifying payoff differences) and replaces the whole population at once, whereas imitation uses the Fermi rule with a moderate β, and the Moran process uses linear (proportional) fitness.

```
import numpy as np
from math import comb

M = np.array([[2, 0], [0, 1]])
N = 4
v1, v2 = 2, 2
beta = 2

# Payoffs
pi1 = ((v1 - 1) * M[0, 0] + v2 * M[0, 1]) / (N - 1)
pi2 = (v1 * M[1, 0] + (v2 - 1) * M[1, 1]) / (N - 1)
print(f"pi_1 = {pi1:.4f}, pi_2 = {pi2:.4f}")

# 1. Imitation
P_2to1 = 1 / (1 + np.exp(-beta * (pi1 - pi2)))
P_imitation_increase = (v2 / N) * (v1 / (N - 1)) * P_2to1
print(f"Imitation P(increase) = {P_imitation_increase:.4f}")

# 2. Moran
total_fitness = v1 * pi1 + v2 * pi2
P_moran_increase = (v1 * pi1 / total_fitness) * (v2 / N)
print(f"Moran P(increase) = {P_moran_increase:.4f}")

# 3. Wright-Fisher
f1 = np.exp(beta * pi1)
f2 = np.exp(beta * pi2)
p1 = (v1 * f1) / (v1 * f1 + v2 * f2)
P_WF_3 = comb(N, 3) * p1**3 * (1 - p1)**1
print(f"Wright-Fisher p_1 = {p1:.4f}")
print(f"Wright-Fisher P(next state = (3,1)) = {P_WF_3:.4f}")
```

```
pi_1 = 0.6667, pi_2 = 0.3333
Imitation P(increase) = 0.2203
Moran P(increase) = 0.3333
Wright-Fisher p_1 = 0.6608
Wright-Fisher P(next state = (3,1)) = 0.3915
```

Solution 12.3: Solution to Exercise 3

The game is $M = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$ with population shares x_1 and $x_2 = 1 - x_1$.

The expected payoffs against the current population are:

$$\pi_1(x_1) = 3x_1 + 0 \cdot (1 - x_1) = 3x_1 \tag{12.27}$$

$$\pi_2(x_1) = 0 \cdot x_1 + 2(1 - x_1) = 2(1 - x_1) \tag{12.28}$$

1. **Rest points of best-response dynamics.**

From the definition of best-response dynamics:

$$\dot{x}_1 = \mathbf{1}[\pi_1 = \max(\pi_1, \pi_2)] - x_1 \tag{12.29}$$

At a rest point $\dot{x}_1 = 0$. Three cases arise:

- If $\pi_1 > \pi_2$ (i.e., $3x_1 > 2(1 - x_1)$, i.e., $x_1 > 2/5$): $\dot{x}_1 = 1 - x_1 = 0 \Rightarrow x_1 = 1$.
- If $\pi_1 < \pi_2$ (i.e., $x_1 < 2/5$): $\dot{x}_1 = 0 - x_1 = 0 \Rightarrow x_1 = 0$.
- If $\pi_1 = \pi_2$ (i.e., $x_1 = 2/5$): both strategies are best responses, so by the convention $\mathbf{1}[\pi_1 = \max] = 1/2$ the target share is $1/2$, giving $\dot{x}_1 = 1/2 - 2/5 = 1/10 \neq 0$.

So $x_1 = 2/5$ is **not** a rest point of the best-response dynamics (in continuous time with this convention).

The rest points are $x_1 \in \{0, 1\}$.

1. **Stability of each rest point.**

   - For $x_1 \in (0, 2/5)$: $\pi_2 > \pi_1$, so $\dot{x}_1 = 0 - x_1 = -x_1 < 0$. The trajectory moves toward $x_1 = 0$. Thus $x_1 = 0$ is **stable**.
   - For $x_1 \in (2/5, 1)$: $\pi_1 > \pi_2$, so $\dot{x}_1 = 1 - x_1 > 0$. The trajectory moves toward $x_1 = 1$. Thus $x_1 = 1$ is **stable**.
   - The threshold $x_1 = 2/5$ is an unstable boundary: perturbations away from $2/5$ are amplified, not corrected.

Both pure states are stable rest points. The basin of attraction of $x_1 = 0$ is $[0, 2/5)$ and of $x_1 = 1$ is $(2/5, 1]$.

1. **Comparison to Nash equilibria.**

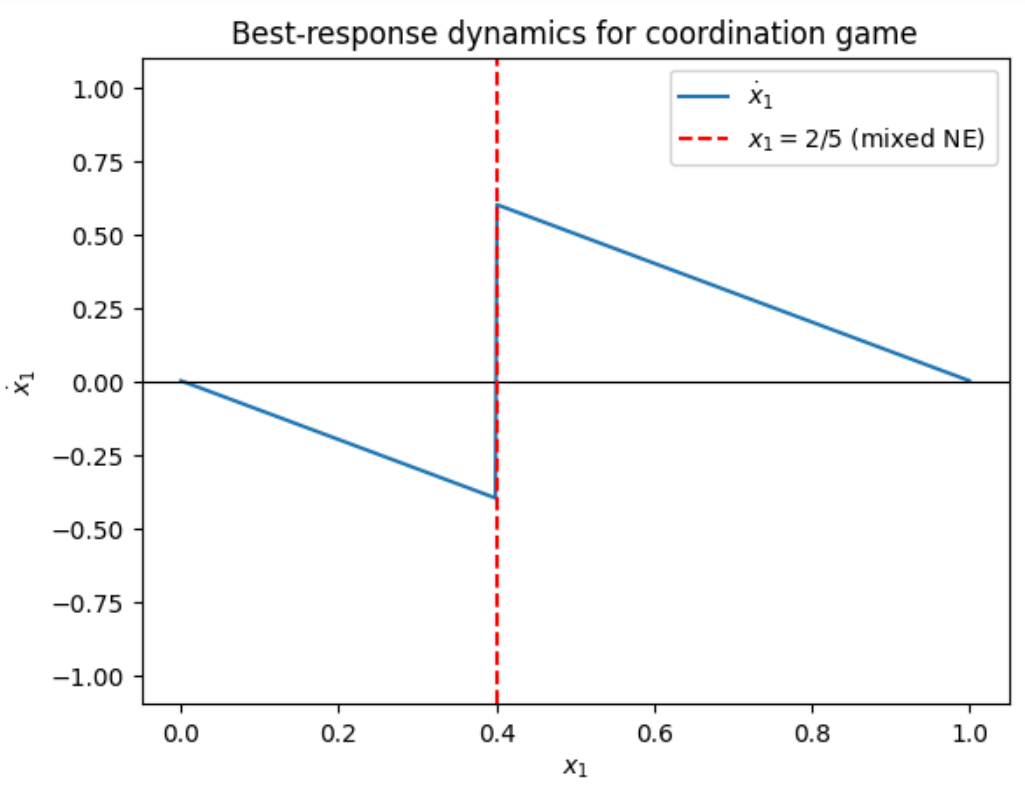
The Nash equilibria of $M = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$ are:

- Pure Nash: $(1, 0)$ and $(0, 1)$ (both strict, as they are the best response to themselves in a coordination game).
- Mixed Nash: $(x_1^*, 1 - x_1^*)$ with $x_1^* = 2/5$, found by making the opponent indifferent: $3x_1 = 2(1 - x_1) \Rightarrow x_1 = 2/5$.

The two stable rest points $x_1 \in \{0, 1\}$ correspond exactly to the two strict Nash equilibria. The mixed Nash equilibrium at $x_1 = 2/5$ is **not** a rest point of the best-response dynamics in the continuous formulation (and is unstable under nearby perturbations). This is consistent with the general result that rest points of best-response dynamics are Nash equilibria, and strict Nash equilibria are always stable rest points.

```
import numpy as np
import matplotlib.pyplot as plt

M = np.array([[3, 0], [0, 2]])
x1_values = np.linspace(0, 1, 500)

pi1 = M[0, 0] * x1_values + M[0, 1] * (1 - x1_values)
pi2 = M[1, 0] * x1_values + M[1, 1] * (1 - x1_values)
```

```
# Best-response dynamics: dx1/dt = BR(x) - x1
# BR = 1 if pi1 > pi2, 0 if pi1 < pi2, 1/2 if equal
BR = np.where(pi1 > pi2, 1.0, np.where(pi1 < pi2, 0.0, 0.5))
dx1_dt = BR - x1_values

plt.figure()
plt.plot(x1_values, dx1_dt, label=r"$\dot{x}_1$")
plt.axhline(0, color="black", linewidth=0.8)
plt.axvline(2/5, color="red", linestyle="--", label=r"$x_1=2/5$ (mixed NE)")
plt.xlabel("$x_1$")
plt.ylabel(r"$\dot{x}_1$")
plt.title("Best-response dynamics for coordination game")
plt.legend()
plt.ylim(-1.1, 1.1);
```



Solution 12.4: Solution to Exercise 4

The Stag Hunt game is $M = \begin{pmatrix} 4 & 0 \\ 1 & 2 \end{pmatrix}$ with x_1 denoting the share of the population playing strategy 1 (Stag).

The expected payoffs are:

$$\pi_1(x_1) = 4x_1 + 0 \cdot (1 - x_1) = 4x_1 \tag{12.30}$$

$$\pi_2(x_1) = 1 \cdot x_1 + 2(1 - x_1) = 2 - x_1 \tag{12.31}$$

1. **Replicator dynamics from $x_1 = 0.4$.**

The average fitness is:

$$\begin{aligned}\bar{\pi} &= x_1\pi_1 + (1-x_1)\pi_2 = x_1 \cdot 4x_1 + (1-x_1)(2-x_1) \\ &= 4x_1^2 + 2 - 3x_1 + x_1^2 = 5x_1^2 - 3x_1 + 2\end{aligned} \tag{12.32}$$

The replicator equation:

$$\begin{aligned}\dot{x}_1 &= x_1(\pi_1 - \bar{\pi}) = x_1(4x_1 - (5x_1^2 - 3x_1 + 2)) \\ &= x_1(-5x_1^2 + 7x_1 - 2) \\ &= -x_1(5x_1 - 2)(x_1 - 1)\end{aligned} \tag{12.33}$$

The interior fixed point where $\pi_1 = \pi_2$:

$$4x_1 = 2 - x_1 \Rightarrow x_1^* = \frac{2}{5} = 0.4 \tag{12.34}$$

At $x_1 = 0.4$ exactly, we are at the interior fixed point $x_1^* = 2/5$. The stability at x_1^* determines where nearby trajectories go.

Evaluate the derivative of $\dot{x}_1$ at x_1^*: for x_1 slightly above $2/5$ (say $x_1 = 0.41$):

$$\pi_1(0.41) = 4(0.41) = 1.64 > \pi_2(0.41) = 2 - 0.41 = 1.59 \tag{12.35}$$

So $\dot{x}_1 > 0$ for $x_1 > 2/5$: the trajectory moves toward $x_1 = 1$.

For x_1 slightly below $2/5$ (say $x_1 = 0.39$):

$$\pi_1(0.39) = 1.56 < \pi_2(0.39) = 1.61 \tag{12.36}$$

So $\dot{x}_1 < 0$ for $x_1 < 2/5$: the trajectory moves toward $x_1 = 0$.

Since the initial condition $x_1 = 0.4 = x_1^*$ is exactly the unstable fixed point, the system is in a degenerate case. Any infinitesimal perturbation above $2/5$ leads to convergence to $x_1 = 1$ (payoff-dominant, Stag equilibrium), and any perturbation below $2/5$ leads to $x_1 = 0$ (risk-dominant, Hare equilibrium).

Starting from $x_1 = 0.4$ exactly, the system remains at this unstable equilibrium in exact arithmetic. In practice, numerical integration will send it to one of the two pure states depending on rounding. The dynamics are exquisitely sensitive at this threshold; this is the defining feature of the mixed Nash equilibrium being unstable under replicator dynamics.

1. **Best-response dynamics: basins of attraction.**

Under best-response dynamics, the best response switches at $\pi_1 = \pi_2$, i.e., at $x_1^* = 2/5$.

- For $x_1 > 2/5$: $\pi_1 > \pi_2$, so the best response is strategy 1. $\dot{x}_1 = 1 - x_1 > 0$. Basin of attraction of $(1, 0)$: $(2/5, 1]$.
- For $x_1 < 2/5$: $\pi_2 > \pi_1$, so best response is strategy 2. $\dot{x}_1 = 0 - x_1 = -x_1 < 0$. Basin of attraction of $(0, 1)$: $[0, 2/5)$.

The basin of the payoff-dominant equilibrium $(1, 0)$ is $(2/5, 1]$. The basin of the risk-dominant equilibrium $(0, 1)$ is $[0, 2/5)$.

Note: the risk-dominant equilibrium $(0, 1)$ has the larger basin under best-response dynamics (since $2/5 < 1/2$). This is in fact a general result: best-response dynamics (and many learning dynamics) favour the risk-dominant equilibrium when the mixed Nash equilibrium lies below $1/2$.

1. **Why the two dynamics can select different equilibria.**

Best-response dynamics and replicator dynamics both have the same rest points (Nash equilibria) and the same stability classification at the pure states. However, they can differ in their transient behaviour and selected equilibrium in two key ways:

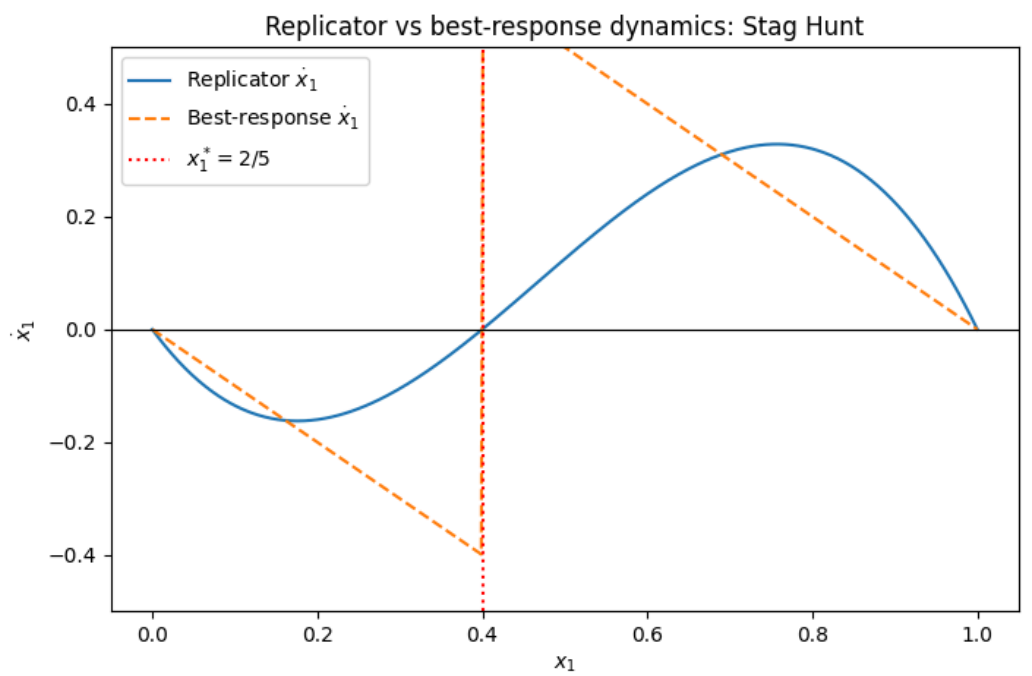
- **Speed of convergence**: Replicator dynamics amplifies payoff differences gradually (proportional to current frequency x_i and excess payoff), while best-response dynamics switches immediately and completely to the best strategy. Near the interior fixed point, the two dynamics can therefore traverse different paths in state space.
- **Selection at the boundary**: In games where the interior equilibrium coincides with a risk-dominant threshold ($x^* \neq 1/2$), the basin of attraction under replicator dynamics depends on the cubic form of the replicator equation, while under best-response dynamics the basin is determined purely by x^* as a linear threshold. For this game both agree (the unstable fixed point is at $x^* = 2/5$ for both), but in more complex multi-strategy games the basins can genuinely differ.
- **Stochastic finite-population versions**: Moran-process-like stochastic analogues of these dynamics can select different equilibria because the transition probabilities depend on the microscopic update rule, not just the mean-field dynamics.

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sym

x_1 = sym.Symbol("x_1")
M_sym = sym.Matrix([[4, 0], [1, 2]])

pi1 = 4 * x_1
pi2 = 2 - x_1
phi = x_1 * pi1 + (1 - x_1) * pi2
x1_dot = x_1 * (pi1 - phi)
x1_dot_simplified = sym.simplify(x1_dot)
print("Replicator equation:", x1_dot_simplified)
fixed_points = sym.solveset(x1_dot_simplified, x_1)
print("Fixed points:", fixed_points)
```

```
Replicator equation: x_1*(-5*x_1**2 + 7*x_1 - 2)
Fixed points: {0, 2/5, 1}
```

```
x1_values = np.linspace(0, 1, 500)
pi1_num = 4 * x1_values
pi2_num = 2 - x1_values
phi_num = x1_values * pi1_num + (1 - x1_values) * pi2_num
x1_dot_num = x1_values * (pi1_num - phi_num)

# Best-response dynamics
BR = np.where(pi1_num > pi2_num, 1.0, np.where(pi1_num < pi2_num, 0.0, 0.5))
br_dot = BR - x1_values

plt.figure(figsize=(8, 5))
plt.plot(x1_values, x1_dot_num, label="Replicator $\\dot{x}_1$")
plt.plot(x1_values, br_dot, label="Best-response $\\dot{x}_1$",
linestyle="--")
plt.axhline(0, color="black", linewidth=0.8)
```

```
plt.axvline(2/5, color="red", linestyle=":", label=r"$x_1^*=2/5$")
plt.xlabel("$x_1$")
plt.ylabel(r"$\dot{x}_1$")
plt.title("Replicator vs best-response dynamics: Stag Hunt")
plt.legend()
plt.ylim(-0.5, 0.5);
```



13. Best response polytopes

The set of mixed strategies that best respond to some opponent strategy forms a polytope, and Nash equilibria correspond to complementary vertices of a pair of such polytopes. This chapter develops this geometric perspective and introduces the Lemke–Howson algorithm for finding Nash equilibria in two-player games.

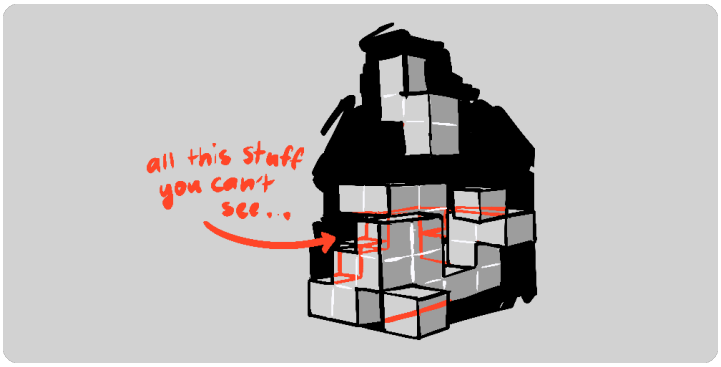


Figure 13.1: Geometric pieces fitting together. The best responses of each player form a polytope, and the Nash equilibria of a game sit at carefully matched vertices of a pair of such shapes.

13.1 Motivating Example: Insider Threat Detection

A large organization is conducting a red team exercise to test its internal security against potential insider threats.

A simulated **Leaker** (the row player) chooses among three methods of exfiltrating sensitive data: **USB drive**, **Personal email**, or **Cloud storage**.

The **Defender** (the column player) allocates monitoring resources to one of three detection strategies: **Endpoint Monitoring**, **Email Filtering**, or **Cloud Auditing**.

A government **regulator** observes the system and wants to assess whether current security measures are adequate. It is assumed that both players act optimally, i.e., they adopt strategies that correspond to a **Nash equilibrium**.

The regulator evaluates the system based on the **equilibrium probability of a successful breach**. If this exceeds a critical threshold, the regulator will mandate additional investment in security.

Let the row player (Leaker) choose among USB, Email, and Cloud, and let the column player (Defender) choose among Endpoint Monitoring, Email Filtering, and Cloud Auditing.

The Leaker's payoff matrix (probabilities of successful exfiltration) is:

$$M_r = \begin{pmatrix} 7/9 & 4/5 & 5/9 \\ 7/10 & 5/8 & 7/8 \\ 3/5 & 4/5 & 5/7 \end{pmatrix} \tag{13.1}$$

The Defender's payoff matrix (probabilities of successful detection) is:

$$M_c = \begin{pmatrix} 1/7 & 1/10 & 2/5 \\ 1/3 & 5/9 & 1/5 \\ 4/9 & 1/3 & 1/4 \end{pmatrix} \tag{13.2}$$

Note that the sum of payoffs in any outcome need not equal 1. This reflects the possibility that the attacker fails to exfiltrate and the defender fails to detect the attempt, for instance, if the file transfer crashes mid-way and no alert is triggered.

This point is also important mathematically: the game is **not constant-sum** and therefore not equivalent to a zero-sum game. It is also **strategically rich**: each player has three actions, none of which can be removed through simple rationalisation. This chapter introduces efficient methods for computing Nash equilibria in such settings.

13.2 Theory

13.2.1 Definition: Best Response Polytopes

For a two player game $(M_r, M_c) \in \mathbb{R}^{m\times n}_{(>0)^2}$ the row/column player best response polytope $\mathcal{P}_r/\mathcal{P}_c$ is defined by:

$$\mathcal{P}_r = \{x \in \mathbb{R}^m \mid x \geq 0; xM_c \leq 1\} \tag{13.3}$$

$$\mathcal{P}_c = \{y \in \mathbb{R}^n \mid M_r y \leq 1; y \geq 0\} \tag{13.4}$$

The polytope $\mathcal{P}_r$, corresponds to the set of points with an upper bound on the utility of those points when considered as row strategies against which the column player plays.

The polytope $\mathcal{P}_c$, corresponds to the set of points with an upper bound on the utility of those points when considered as column strategies against which the row player plays.

> **Note**
>
> The fact that these polytopes are defined for $M_r, M_c > 0$ is not restrictive in practice as we can add a constant to our utilities.

13.2.1.1 Example: Best Response Polytopes for the threat detection game

Let us construct the best response polytopes for the threat detection game.

Applying the definition (13.3) we have:

$$\mathcal{P}_r = \left\{ x \in \mathbb{R}^3 \mid x_i \geq 0 \text{ and } \left(x \begin{pmatrix} 1/7 & 1/10 & 2/5 \\ 1/3 & 5/9 & 1/5 \\ 4/9 & 1/3 & 1/4 \end{pmatrix} \right)_{ij} \leq 1 \quad \text{for all } 1 \leq i, j \leq 3 \right\} \tag{13.5}$$

$$\mathcal{P}_c = \left\{ y \in \mathbb{R}^3 \mid \left(\begin{pmatrix} 7/9 & 4/5 & 5/9 \\ 7/10 & 5/8 & 7/8 \\ 3/5 & 4/5 & 5/7 \end{pmatrix} y \right)_{ij} \leq 1 \quad \text{and } y_j \geq 0 \quad \text{for all } 1 \leq i, j \leq 3 \right\} \tag{13.6}$$

This gives:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \\ x_1/7 + x_2/3 + 4x_3/9 &\leq 1 \\ x_1/10 + 5x_2/9 + x_3/3 &\leq 1 \\ 2x_1/5 + x_2/5 + x_3/4 &\leq 1 \end{aligned} \tag{13.7}$$

$$
\begin{aligned}
7y_1/9 + 4y_2/5 + 5y_3/9 &\leq 1 \\
7y_1/10 + 5y_2/8 + 7y_3/8 &\leq 1 \\
3y_1/5 + 4y_2/5 + 5y_3/7 &\leq 1 \\
y_1 &\geq 0 \\
y_2 &\geq 0 \\
y_3 &\geq 0
\end{aligned}
\tag{13.8}
$$

Important

The ordering of these two sets of inequalities is important. The non-negativity constraints are first for P_r and second for P_c. This will be used in Definition: Vertex Labelling.

The vertices of these two polytopes are:

For $\mathcal{P}_r$:

$$
\begin{aligned}
v_0 &= (0, 0, 0) \\
v_1 &= (5/2, 0, 0) \\
v_2 &= (0, 0, 9/4) \\
v_3 &= (245/179, 0, 324/179) \\
v_4 &= (160/91, 135/91, 0) \\
v_5 &= (6580/4927, 4185/4927, 5832/4927) \\
v_6 &= (0, 9/5, 0) \\
v_7 &= (0, 9/11, 18/11)
\end{aligned}
\tag{13.9}
$$

These are shown in Figure 13.2.

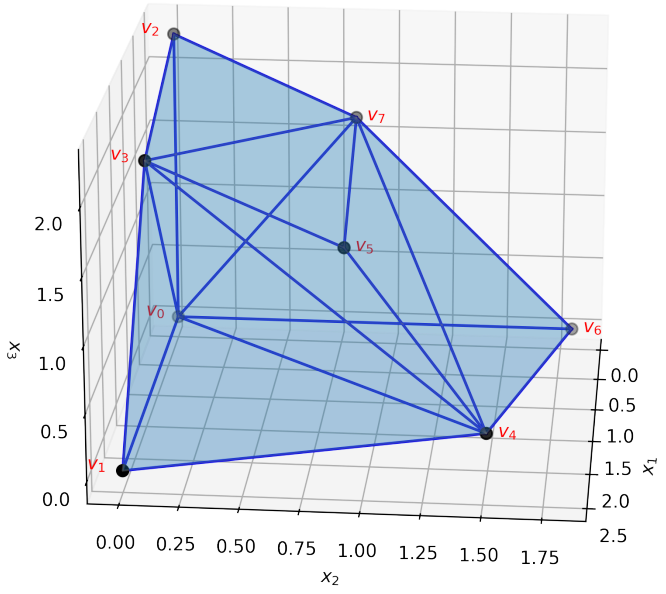


Figure 13.2: The three dimensional $\mathcal{P}_r$.

For $\mathcal{P}_c$:

$$
\begin{aligned}
u_0 &= (0, 0, 0) \\
u_1 &= (9/7, 0, 0) \\
u_2 &= (0, 0, 8/7) \\
u_3 &= (23/21, 0, 4/15) \\
u_4 &= (0, 45/71, 49/71) \\
u_5 &= (25/67, 40/67, 28/67) \\
u_6 &= (0, 5/4, 0)
\end{aligned}
\tag{13.10}
$$

These are shown in Figure 13.3.

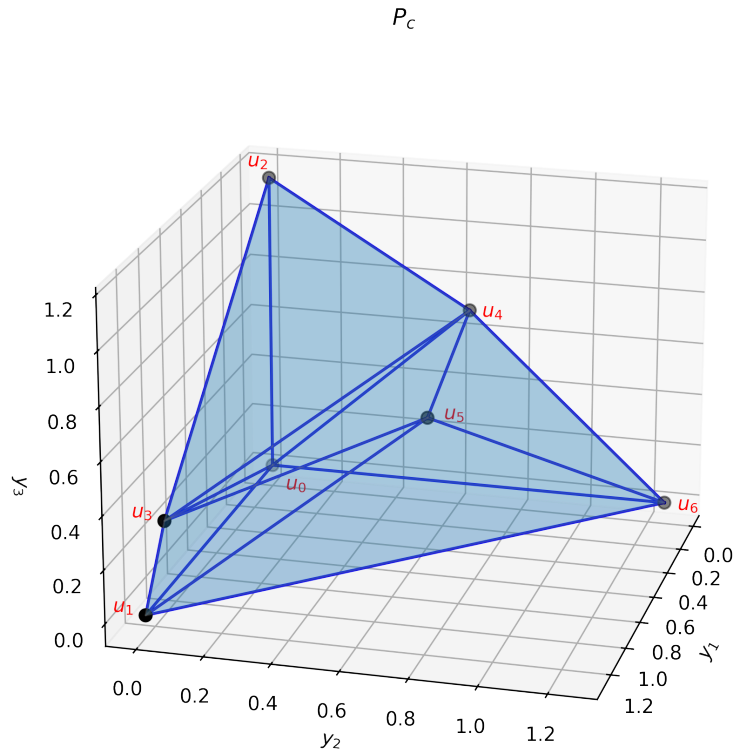


Figure 13.3: The three dimensional $\mathcal{P}_c$.

Note

Vertices of these polytopes are not probability vectors so they are not strategies.

A vertex of a best response polytope corresponds to a strategy through normalization that ensures the sum corresponds to one.

For example u_6 corresponds to a strategy:

$$
\sigma = \frac{u_6}{\sum_{i=1}^{3} u_{6_i}} = (0, 1, 0) \tag{13.11}
$$

Similarly for u_5:

$$
\begin{aligned}
\sigma &= \frac{u_5}{\sum_{i=1}^{3} u_{5_i}} \\
&= \frac{(25/67, 40/67, 28/67)}{(25/67 + 40/67 + 28/67)} \\
&= \frac{(25/67, 40/67, 28/67)}{93/67} \\
&= (25/93, 40/93, 28/93)
\end{aligned} \tag{13.12}
$$

13.2.2 Definition: Vertex Labelling

A **vertex labelling** is an assignment of labels to each vertex of a best response polytope, where each label corresponds to a constraint that is **binding** (i.e., holds with equality) at that vertex.

Each defining inequality of a best response polytope has a game theoretic interpretation when it is a binding inequality for a given vertex.

13.2.2.1 Example: Vertex labelling for the threat detection game

Let us consider the inequalities of P_r and interpret what is implied when the inequality is binding:

$$
\begin{aligned}
x_1 = 0 &\implies \text{textcircled } 1 : \text{the first action is not played by the strategy represented by } x \\
x_2 = 0 &\implies \text{textcircled } 2 : \text{the second action is not played by the strategy represented by } x \\
x_3 = 0 &\implies \text{textcircled } 3 : \text{the third action is not played by the strategy represented by } x \\
(xM_c)_1 = 1 &\implies \text{textcircled } 4 : \text{the first action is a best response to the strategy represented by } x \\
(xM_c)_2 = 1 &\implies \text{textcircled } 5 : \text{the second action is a best response to the strategy represented by } x \\
(xM_c)_3 = 1 &\implies \text{textcircled } 6 : \text{the third action is a best response to the strategy represented by } x
\end{aligned} \tag{13.13}
$$

Similarly for $\mathcal{P}_c$

$$
\begin{aligned}
(yM_r)_1 = 1 &\implies \text{textcircled } 1 : \text{the first action is a best response to the strategy represented by } y \\
(yM_r)_2 = 1 &\implies \text{textcircled } 2 : \text{the second action is a best response to the strategy represented by } y \\
(yM_r)_3 = 1 &\implies \text{textcircled } 3 : \text{the third action is a best response to the strategy represented by } y \\
y_1 = 0 &\implies \text{textcircled } 4 : \text{the first action is not played by the strategy represented by } y \\
y_2 = 0 &\implies \text{textcircled } 5 : \text{the second action is not played by the strategy represented by } y \\
y_3 = 0 &\implies \text{textcircled } 6 : \text{the third action is not played by the strategy represented by } y
\end{aligned} \tag{13.14}
$$

We have used textcircled 1 through textcircled 6 as the labels.

Let us label each of the vertices:

For $v_4 = (160/91, 135/91, 0)$:

$$
\begin{aligned}
x_1 = 160/91 \neq 0 \\
x_2 = 135/91 \neq 0 \\
x_3 = 0 &\implies \text{textcircled } 3 : \text{the third action is not played by the strategy represented by } x \\
(xM_c)_1 = 475/637
\end{aligned} \tag{13.15}
$$

$$
\begin{aligned}
(xM_c)_2 = 1 &\implies \text{textcircled } 5 : \text{the second action is a best response to the strategy represented by } x \\
(xM_c)_3 = 1 &\implies \text{textcircled } 6 : \text{the third action is a best response to the strategy represented by } x
\end{aligned}
$$

For $\mathcal{P}_r$:

$$\begin{aligned}
\mathcal{L}(v_0) &= \mathcal{L}((0,0,0)) = \{1,2,3\} \\
\mathcal{L}(v_1) &= \mathcal{L}((5/2,0,0)) = \{2,3,6\} \\
\mathcal{L}(v_2) &= \mathcal{L}((0,0,9/4)) = \{1,2,4\} \\
\mathcal{L}(v_3) &= \mathcal{L}((245/179,0,324/179)) = \{2,4,6\} \\
\mathcal{L}(v_4) &= \mathcal{L}((160/91,135/91,0)) = \{3,5,6\} \\
\mathcal{L}(v_5) &= \mathcal{L}((6580/4927,4185/4927,5832/4927)) = \{4,5,6\} \\
\mathcal{L}(v_6) &= \mathcal{L}((0,9/5,0)) = \{1,3,5\} \\
\mathcal{L}(v_7) &= \mathcal{L}((0,9/11,18/11)) = \{1,4,5\}
\end{aligned} \tag{13.16}$$

For $\mathcal{P}_c$:

$$\begin{aligned}
\mathcal{L}(u_0) &= \mathcal{L}((0,0,0)) = \{4,5,6\} \\
\mathcal{L}(u_1) &= \mathcal{L}((9/7,0,0)) = \{1,5,6\} \\
\mathcal{L}(u_2) &= \mathcal{L}((0,0,8/7)) = \{2,4,5\} \\
\mathcal{L}(u_3) &= \mathcal{L}((23/21,0,4/15)) = \{1,2,5\} \\
\mathcal{L}(u_4) &= \mathcal{L}((0,45/71,49/71)) = \{2,3,4\} \\
\mathcal{L}(u_5) &= \mathcal{L}((25/67,40/67,28/67)) = \{1,2,3\} \\
\mathcal{L}(u_6) &= \mathcal{L}((0,5/4,0)) = \{1,3,4,6\}
\end{aligned} \tag{13.17}$$

Important

We can use labels to identify if pairs of strategies are best responses to each other.

If a label is present in either vertex then either of the following are true:

- it is indicating that an action is not played (for example label 1 in $\mathcal{P}_r$).
- it is indicating that the same action is a best response to the strategy represented by the vector (for example label 1 in $\mathcal{P}_c$).

Looking at v_5 and u_5 the union of the labels of these vertices gives the full set of vertices.

This leads to the following definition.

13.2.3 Definition: Fully labelled vertex pair

A pair of vertices $(x, y) \in \mathcal{P}_r \times \mathcal{P}_c$ is said to be a **fully labelled vertex pair** if the union of their labels covers all possible labels:

$$\mathcal{L}(x) \cup \mathcal{L}(y) = \{1, 2, ..., m+n\} \tag{13.18}$$

Such a pair, when normalised (so that the components sum to 1), corresponds to a Nash equilibrium.

13.2.3.1 Example: Fully labelled vertex pair for the threat detection game

As shown in Example: Vertex labelling for the threat detection game:

$$v_5 \cup u_5 = \{4,5,6\} \cup \{1,2,3\} = \{1,2,3,4,5,6\} \tag{13.19}$$

is a fully labelled vertex pair.

This corresponds to the normalised strategies:

$$\sigma_r = \frac{(6580/4927, 4185/4927, 5832/4927)}{6580/4927 + 4185/4927 + 5832/4927} = (940/2371, 4185/16597, 5832/16597) \quad (13.20)$$

and

$$\sigma_c = \frac{(25/67, 40/67, 28/67)}{(25/67 + 40/67 + 28/67)} = (25/67, 40/67, 28/67) \quad (13.21)$$

which is a Nash equilibrium.

Searching through all pairs of vertices is one approach to identifying Nash equilibria although a more efficient approach will now be discussed.

13.2.4 Definition: Lemke-Howson Algorithm

For a nondegenerate 2 player game $(M_r, M_c) \in \mathbb{R}^{m \times n}_{(>0)^2}$ the following algorithm returns a Nash equilibrium:

1. Start at the artificial equilibrium: $(0, 0)$
2. Choose a label to drop.
3. Remove this label from the corresponding vertex by traversing an edge of the corresponding polytope to another vertex.
4. The new vertex will now have a duplicate label in the other polytope. Remove this label from the vertex of the other polytope and traverse an edge of that polytope to another vertex.
5. Repeat step 4 until the pair of vertices is fully labelled.

13.2.4.1 Example: Application of the Lemke–Howson algorithm for the threat detection game with known vertices and labels

We will use Figure 13.2,
Figure 13.3, as well as
(13.16) and (13.17) to move
from vertex to vertex in the threat detection game.

We apply the algorithm as follows:

1. Start at (v_0, u_0) and choose to drop label 1 (an arbitrary choice).
2. Label 1 is not among the labels of u_0, so we move from v_0 to an adjacent vertex (v_1, v_2, v_3, v_4, v_6, or v_7) that does not carry label 1 but shares other labels with v_0. We select v_1. This introduces label 6, which must now be dropped in $\mathcal{P}_c$.
3. $(v_1, u_0) \to (v_1, u_2)$: the labels are $\{2, 3, 6\}, \{2, 4, 5\}$. Label 2 must be dropped in $\mathcal{P}_r$.
4. $(v_1, u_2) \to (v_4, u_2)$: the labels are $\{3, 5, 6\}, \{2, 4, 5\}$. Label 5 must be dropped in $\mathcal{P}_c$.
5. $(v_4, u_2) \to (v_4, u_4)$: the labels are $\{3, 5, 6\}, \{2, 3, 4\}$. Label 3 must be dropped in $\mathcal{P}_r$.
6. $(v_4, u_4) \to (v_5, u_4)$: the labels are $\{4, 5, 6\}, \{2, 3, 4\}$. Label 4 must be dropped in $\mathcal{P}_c$.
7. $(v_5, u_4) \to (v_5, u_5)$: the labels are $\{4, 5, 6\}, \{1, 2, 3\}$. This is a fully labelled vertex pair.

After normalisation, this yields the Nash equilibrium computed in
Example: Fully labelled vertex pair for the threat detection game.

This approach, while systematic, is only efficient here because the vertices have already been computed. In practice, obtaining the vertices of the polytope can be a time-consuming process. In the next example, we will demonstrate how the Lemke–Howson algorithm becomes truly efficient through integer pivoting.

13.2.4.2 Example: Application of the Lemke-Howson Algorithm for the threat detection game with integer pivoting

Using the definition of a tableau the tableaux for a 2 player game $(M_r, M_c) \in \mathbb{R}^{m\times n}_{(>0)^2}$ are given by:

$$T_r = (B^T \;\; I \;\; 1) \tag{13.22}$$

and

$$T_c = (I \;\; A \;\; 1) \tag{13.23}$$

In the case of the threat detection game this gives:

$$T_r = \begin{pmatrix} 1/7 & 1/3 & 4/9 & 1 & 0 & 0 & 1 \\ 1/10 & 5/9 & 1/3 & 0 & 1 & 0 & 1 \\ 2/5 & 1/5 & 1/4 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{13.24}$$

and

$$T_c = \begin{pmatrix} 1 & 0 & 0 & 7/9 & 4/5 & 5/9 & 1 \\ 0 & 1 & 0 & 7/10 & 5/8 & 7/8 & 1 \\ 0 & 0 & 1 & 3/5 & 4/5 & 5/7 & 1 \end{pmatrix} \tag{13.25}$$

Important

The non basic columns correspond to labels.

Let us now reproduce the steps of Example: Application of the Lemke–Howson algorithm for the threat detection game with known vertices and labels: we begin by dropping label 1 which corresponds to the non basic variable 1 of T_r. In terms of integer pivoting this is done by pivoting the first column of T_r.

As described in Definition: Integer Pivoting we carry out the minimum ratio test:

1. The ratio for the first row: $\frac{7}{1}$
2. The ratio for the second row: $\frac{10}{1}$
3. The ratio for the third row: $\frac{5}{2}$

We pivot on the third row giving:

$$T_r = \begin{pmatrix} 0 & 11/105 & 179/1260 & 2/5 & 0 & -1/7 & 9/35 \\ 0 & 91/450 & 13/120 & 0 & 2/5 & -1/10 & 3/10 \\ 1 & 1/2 & 5/8 & 0 & 0 & 5/2 & 5/2 \end{pmatrix} \tag{13.26}$$

This tableau has labels/non basic variables $\{2, 3, 6\}$. So we now pivot column 6 in T_c. The minimum ratio test:

1. The ratio for the first row: $\frac{9}{5}$
2. The ratio for the second row: $\frac{8}{7}$
3. The ratio for the third row: $\frac{7}{5}$

We pivot on the second row giving:

$$T_c = \begin{pmatrix} 7/8 & -5/9 & 0 & 7/24 & 127/360 & 0 & 23/72 \\ 0 & 8/7 & 0 & 4/5 & 5/7 & 1 & 8/7 \\ 0 & -5/7 & 7/8 & 1/40 & 71/280 & 0 & 9/56 \end{pmatrix} \tag{13.27}$$

This tableau has labels $\{2, 4, 5\}$ so we pivot column 2 in T_r. The minimum ratio test:

1. The ratio for the first row: $\frac{9/35}{11/105} = 27/11$

2. The ratio for the second row: $\frac{3/10}{91/450} = 135/91$
3. The ratio for the third row: $\frac{5/2}{1/2} = 5/2$

We pivot on the second row giving:

$$T_r = \begin{pmatrix} 0 & 0 & 4927/283500 & 91/1125 & -22/525 & -29/1575 & 18/875 \\ 0 & 1 & 15/28 & 0 & 180/91 & -45/91 & 135/91 \\ 91/450 & 0 & 13/180 & 0 & -1/5 & 5/9 & 16/45 \end{pmatrix} \tag{13.28}$$

This tableau has labels $\{3, 5, 6\}$ so we pivot column 5 in T_c. The minimum ratio test:

1. The ratio for the first row: $\frac{23/72}{127/360} = 115/127$
2. The ratio for the second row: $\frac{8/7}{1} = 8/7$
3. The ratio for the third row: $\frac{9/56}{71/280} = 45/71$

We pivot on the third row giving:

$$T_c = \begin{pmatrix} 71/320 & 1/9 & -889/2880 & 469/7200 & 0 & 0 & 7/288 \\ 0 & 4/5 & -5/8 & 37/200 & 0 & 71/280 & 7/40 \\ 0 & -200/71 & 245/71 & 7/71 & 1 & 0 & 45/71 \end{pmatrix} \tag{13.29}$$

This tableau has labels $\{2, 3, 4\}$ so we pivot column 3 in T_r. The minimum ratio test:

1. The ratio for the first row: $\frac{18/875}{4927/283500} = 5832/4927$
2. The ratio for the second row: $\frac{135/91}{15/28} = 524/195$
3. The ratio for the third row: $\frac{16/45}{13/180} = 576/65$

We pivot on the first row giving:

$$T_r = \begin{pmatrix} 0 & 0 & 1 & 1764/379 & -11880/4927 & -5220/4927 & 5832/4927 \\ 0 & 4927/283500 & 0 & -13/300 & 179/3150 & 2/1575 & 31/2100 \\ 64051/18225000 & 0 & 0 & -1183/202500 & -91/202500 & 1001/91125 & 4277/911250 \end{pmatrix}$$

This tableau has labels $\{4, 5, 6\}$ so we pivot column 4 in T_c. The minimum ratio test:

1. The ratio for the first row: $\frac{7/288}{469/7200} = 25/67$
2. The ratio for the second row: $\frac{7/40}{37/200} = 35/37$
3. The ratio for the third row: $\frac{45/71}{7/71} = 45/7$

We pivot on the first row giving:

$$T_c = \begin{pmatrix} 3195/938 & 800/469 & -635/134 & 1 & 0 & 0 & 25/67 \\ -2627/64000 & 71/2250 & 9443/576000 & 0 & 0 & 4757/288000 & 497/72000 \\ -7/320 & -7/36 & 49/192 & 0 & 469/7200 & 0 & 7/180 \end{pmatrix} \tag{13.31}$$

This tableau has labels $\{1, 2, 3\}$ so we have a fully labelled vertex pair.

Setting the non-basic variables to 0 we have the following systems of equations:

$$\begin{aligned} 64051/18225000x_1 &= 4277/911250 \\ 4927/283500x_2 &= 31/2100 \\ x_3 &= 5832/4927 \end{aligned} \tag{13.32}$$

and:

$$\begin{aligned} y_1 &= 25/67 \\ 469/7200y_2 &= 7/180 \\ 4757/28800y_3 &= 497/7200 \end{aligned} \tag{13.33}$$

Giving: $x = (6580/4927, 4185/4927, 5832/4927)$ and $y = (25/67, 40/67, 28/67)$ which when normalised (13.20) - (13.21) gives:

$$\sigma_r = (940/2371, 4185/16597, 5832/16597) \qquad \sigma_c = (25/93, 40/93, 28/93) \tag{13.34}$$

13.3 Exercises

Exercise 13.1:

For each of the following games, draw the best response polytopes and identify all fully labelled vertex pairs:

1. $$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{13.35}$$

1. $$A = \begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} -2 & 2 \\ 1 & -2 \end{pmatrix} \tag{13.36}$$

Exercise 13.2:

Using the games from Exercise 1, apply the Lemke–Howson algorithm to compute a Nash equilibrium in each case. Carry out the algorithm twice: once using the known vertex structure, and once using integer pivoting.

Exercise 13.3:

In a busy university district, two independent coffee shops compete for customers. Each can choose to:

- **Lower prices** to attract bargain-seekers,
- **Improve quality** by investing in better beans and baristas,
- **Advertise** aggressively through social media and flyers.

Customers respond differently depending on the overall landscape of offers. The three main customer profiles are:

- **Students** who are price-sensitive,
- **Coffee aficionados** who value taste,
- **Impulse buyers** who respond to advertising.

The payoff matrices, based on estimated profits (row player: Café A, column player: Café B), are:

$$M_r = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 4 & 1 \\ 1 & 3 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 2 & 4 \\ 5 & 1 & 3 \end{pmatrix} \tag{13.37}$$

1. Use the Lemke–Howson algorithm to find a Nash equilibrium. Describe how different choices of dropped label may lead to different paths.

2. Interpret the equilibrium in terms of business strategy: What might the equilibrium suggest about the balance between pricing, quality, and advertising?

Exercise 13.4:

Assume the game is **nondegenerate**: each vertex of the best response polytopes has exactly the number of labels required to define it, and no label appears more than once across a vertex pair. Also assume that, when applying the Lemke–Howson algorithm, dropping a label from any fully labelled vertex pair always leads to a **distinct** fully labelled vertex pair.

Under these assumptions, prove that the number of fully labelled vertex pairs, and hence the number of Nash equilibria, is always **odd**.

13.4 Programming

13.4.1 Vertex enumeration with Nashpy

Nashpy can be used to generate the polytopes and enumerate all pairs of vertices.

```
import nashpy as nash
import numpy as np


M_r = np.array([
    [3, 1, 2],
    [2, 4, 1],
    [1, 3, 0]
])

M_c = np.array([
    [2, 3, 1],
    [1, 2, 4],
    [5, 1, 3]
])

game = nash.Game(M_r, M_c)
print(list(game.vertex_enumeration()))
```

```
[(array([0.5, 0.5, 0. ]), array([-2.42861287e-17,  2.50000000e-01,
7.50000000e-01]))]
```

13.4.2 Lemke-Howson with Nashpy

Nashpy can be used to carry out the Lemke-Howson algorithm, an efficient way to find a Nash equilibrium.

```
label_to_drop = 0
print(f"Nash equilibrium:
{game.lemke_howson(initial_dropped_label=label_to_drop)}")
```

```
Nash equilibrium: (array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75]))
```

You can also enumerate all possible dropped labels:

```
print(list(game.lemke_howson_enumeration()))
```

```
[(array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])), (array([0.5, 0.5, 0. ]),
array([0.  , 0.25, 0.75])), (array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])),
(array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])), (array([0.5, 0.5, 0. ]),
array([0.  , 0.25, 0.75])), (array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75]))]
```

13.4.3 Vertex enumeration with Gambit

Gambit can be used to enumerate all pairs of vertices:

```
import pygambit as gbt

game = gbt.Game.from_arrays(M_r, M_c)
print(gbt.nash.enummixed_solve(game))
```

```
NashComputationResult(method='enummixed', rational=True, use_strategic=True,
equilibria=[[[Rational(1, 2), Rational(1, 2), Rational(0, 1)], [Rational(0, 1),
Rational(1, 4), Rational(3, 4)]]], parameters={})
```

13.4.4 Lemke-Howson with Gambit

Gambit can be used to carry out the Lemke-Howson algorithm:

```
print(gbt.nash.lcp_solve(game))
```

```
NashComputationResult(method='lcp', rational=True, use_strategic=True,
equilibria=[[[Rational(1, 2), Rational(1, 2), Rational(0, 1)], [Rational(0, 1),
Rational(1, 4), Rational(3, 4)]]], parameters={'stop_after': None, 'max_depth':
None})
```

13.5 Notable Research

The original paper presenting the Lemke–Howson algorithm for two-player games is (Lemke & Howson, 1964). That paper also contains a constructive proof that, in nondegenerate games, the number of Nash equilibria is always Exercise 4. The algorithm was later extended to N-player games in (Wilson, 1971), where the oddness result is also generalised. An alternative proof of the oddness theorem is provided in (Harsanyi, 1973) the author of which was awarded the Nobel prize with Nash and Selten in 1994.

The worst-case complexity of the Lemke–Howson algorithm is analysed in (Savani & Von Stengel, 2004), which demonstrates that the algorithm may require exponential time on specific inputs. **Computing Nash equilibria is a computationally challenging task**; this intuitive difficulty is formalised in (Daskalakis et al., 2009), where the problem is shown to be PPAD-complete, placing it in a class of problems believed not to admit polynomial-time solutions.

13.5.1 Conclusion

This chapter introduced the geometric and algorithmic structure underlying Nash equilibrium computation through the lens of **best response polytopes**. By framing strategy sets as polytopes and interpreting labels as binding constraints, we gain powerful visual and computational tools for equilibrium analysis.

The **Lemke–Howson algorithm** provides a systematic method for tracing paths through these polytopes to identify **fully labelled vertex pairs**, which correspond to Nash equilibria. Though

simple in low dimensions, this method scales to more complex games using tableau-based pivoting techniques such as **integer pivoting**.

Understanding the polyhedral structure of best responses not only aids in computational efficiency but also provides conceptual clarity: each equilibrium arises from a delicate balance of incentives, visible in the geometry of the feasible region.

Table 13.1 summarises the key concepts introduced in this chapter.

Concept	Description
Best response polytope	A polyhedron defined by inequalities corresponding to best response conditions.
Binding inequality	A constraint that holds with equality at a vertex; gives rise to a label.
Vertex labelling	A mapping from vertices to labels indicating inactive actions or best responses.
Fully labelled vertex pair	A pair of vertices whose labels cover all $m + n$ actions, corresponds to a Nash equilibrium.
Lemke–Howson algorithm	A path-following method that constructs equilibria by dropping and replacing labels.
Integer pivoting	A tableau-based pivoting method used to trace Lemke–Howson paths efficiently.

Table 13.1: Summary of best response polytopes

Important

In every nondegenerate two-player game, each Nash equilibrium corresponds to a fully labelled vertex pair of the best response polytopes.

Crucially, we can find such a vertex pair without explicitly constructing the polytopes, using integer pivoting and the Lemke–Howson algorithm.

13.6 Solutions

Solution 13.1: Solution to Exercise 1

For a 2×2 game (A, B) we must first shift the matrices so that all entries are strictly positive (required by the definition of best response polytopes), then construct $\mathcal{P}_r$ and $\mathcal{P}_c$.

1. Game:

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{13.38}$$

Add a constant to make all entries positive. Add 2 to A and 7 to B:

$$A' = \begin{pmatrix} 5 & 1 \\ 4 & 9 \end{pmatrix}, \qquad B' = \begin{pmatrix} 4 & 8 \\ 8 & 1 \end{pmatrix} \tag{13.39}$$

The best response polytopes are:

$$\mathcal{P}_r = \{x \in \mathbb{R}^2 \texttt{ mid } x \geq 0,\, xB' \leq \mathbf{1}\} \tag{13.40}$$

$$\mathcal{P}_c = \{y \in \mathbb{R}^2 \texttt{ mid } A'y \leq \mathbf{1},\, y \geq 0\} \tag{13.41}$$

Writing out the inequalities for $\mathcal{P}_r$ (labels (1)–(4)):

$$\begin{aligned} x_1 &\geq 0 \quad (1)\\ x_2 &\geq 0 \quad (2)\\ 4x_1 + 8x_2 &\leq 1 \quad (3)\\ 8x_1 + x_2 &\leq 1 \quad (4) \end{aligned} \tag{13.42}$$

Writing out the inequalities for $\mathcal{P}_c$ (labels (1)–(4)):

$$\begin{aligned} 5y_1 + y_2 &\leq 1 \quad (1)\\ 4y_1 + 9y_2 &\leq 1 \quad (2)\\ y_1 &\geq 0 \quad (3)\\ y_2 &\geq 0 \quad (4) \end{aligned} \tag{13.43}$$

Vertices of $\mathcal{P}_r$ (each vertex satisfies 2 of the 4 inequalities with equality):

- $v_0 = (0, 0)$: labels $\{1, 2\}$
- $v_1 = (1/8, 0)$: binding (2) and (4); labels $\{2, 4\}$
- $v_2 = (0, 1/8)$: binding (1) and (3); labels $\{1, 3\}$
- v_3: intersection of $4x_1 + 8x_2 = 1$ and $8x_1 + x_2 = 1$.

Solving: from the second equation $x_2 = 1 - 8x_1$. Substituting: $4x_1 + 8(1 - 8x_1) = 1 \Rightarrow 4x_1 + 8 - 64x_1 = 1 \Rightarrow -60x_1 = -7 \Rightarrow x_1 = 7/60$. Then $x_2 = 1 - 8(7/60) = 1 - 56/60 = 4/60 = 1/15$.

$v_3 = (7/60, 1/15)$: labels $\{3, 4\}$.

Vertices of $\mathcal{P}_c$ (each vertex satisfies 2 of the 4 inequalities with equality):

- $u_0 = (0, 0)$: labels $\{3, 4\}$
- $u_1 = (1/5, 0)$: binding (1) ($5(1/5) + 0 = 1$) and (4); labels $\{1, 4\}$
- $u_2 = (0, 1/9)$: binding (2) ($9(1/9) = 1$) and (3); labels $\{2, 3\}$
- u_3: intersection of $5y_1 + y_2 = 1$ and $4y_1 + 9y_2 = 1$.

From first: $y_2 = 1 - 5y_1$. Substituting: $4y_1 + 9(1 - 5y_1) = 1 \Rightarrow 4y_1 + 9 - 45y_1 = 1 \Rightarrow -41y_1 = -8 \Rightarrow y_1 = 8/41$. Then $y_2 = 1 - 40/41 = 1/41$.

$u_3 = (8/41, 1/41)$: labels $\{1, 2\}$.

Finding fully labelled vertex pairs.

We need pairs (v_i, u_j) such that $\mathcal{L}(v_i) \cup \mathcal{L}(u_j) = \{1, 2, 3, 4\}$.

Pair	$\mathcal{L}(v)$	$\mathcal{L}(u)$	Union	Fully labelled?
(v_0, u_0)	$\{1, 2\}$	$\{3, 4\}$	$\{1, 2, 3, 4\}$	Yes
(v_1, u_3)	$\{2, 4\}$	$\{1, 2\}$	$\{1, 2, 4\}$	No
(v_2, u_1)	$\{1, 3\}$	$\{1, 4\}$	$\{1, 3, 4\}$	No
(v_3, u_3)	$\{3, 4\}$	$\{1, 2\}$	$\{1, 2, 3, 4\}$	Yes
(v_1, u_2)	$\{2, 4\}$	$\{2, 3\}$	$\{2, 3, 4\}$	No
(v_2, u_3)	$\{1, 3\}$	$\{1, 2\}$	$\{1, 2, 3\}$	No

Pair	$\mathcal{L}(v)$	$\mathcal{L}(u)$	Union	Fully labelled?
(v_3, u_1)	$\{3,4\}$	$\{1,4\}$	$\{1,3,4\}$	No
(v_3, u_2)	$\{3,4\}$	$\{2,3\}$	$\{2,3,4\}$	No

The fully labelled vertex pairs are (v_0, u_0) and (v_3, u_3).

$(v_0, u_0) = ((0,0),(0,0))$ is the trivial artificial equilibrium.

$(v_3, u_3) = ((7/60, 1/15), (8/41, 1/41))$ corresponds to the Nash equilibrium obtained by normalising:

$$\sigma_r = \frac{(7/60, 1/15)}{7/60 + 4/60} = \frac{(7/60, 4/60)}{11/60} = (7/11, 4/11) \tag{13.44}$$

$$\sigma_c = \frac{(8/41, 1/41)}{9/41} = (8/9, 1/9) \tag{13.45}$$

---

2. Game:

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} -2 & 2 \\ 1 & -2 \end{pmatrix} \tag{13.46}$$

Add 2 to A and 3 to B:

$$A' = \begin{pmatrix} 4 & 1 \\ 3 & 5 \end{pmatrix}, \qquad B' = \begin{pmatrix} 1 & 5 \\ 4 & 1 \end{pmatrix} \tag{13.47}$$

Inequalities for $\mathcal{P}_r$ (labels (1)–(4)):

$$\begin{aligned} x_1 &\geq 0 \quad (1) \\ x_2 &\geq 0 \quad (2) \\ x_1 + 4x_2 &\leq 1 \quad (3) \\ 5x_1 + x_2 &\leq 1 \quad (4) \end{aligned} \tag{13.48}$$

Inequalities for $\mathcal{P}_c$ (labels (1)–(4)):

$$\begin{aligned} 4y_1 + y_2 &\leq 1 \quad (1) \\ 3y_1 + 5y_2 &\leq 1 \quad (2) \\ y_1 &\geq 0 \quad (3) \\ y_2 &\geq 0 \quad (4) \end{aligned} \tag{13.49}$$

Vertices of $\mathcal{P}_r$:

- $v_0 = (0,0)$: labels $\{1,2\}$
- $v_1 = (1/5, 0)$: labels $\{2,4\}$
- $v_2 = (0, 1/4)$: labels $\{1,3\}$
- v_3: intersection of $x_1 + 4x_2 = 1$ and $5x_1 + x_2 = 1$:

From the first: $x_1 = 1 - 4x_2$. Substituting: $5(1-4x_2) + x_2 = 1 \Rightarrow 5 - 19x_2 = 1 \Rightarrow x_2 = 4/19$. Then $x_1 = 1 - 16/19 = 3/19$.

$v_3 = (3/19, 4/19)$: labels $\{3,4\}$.

Vertices of $\mathcal{P}_c$:

- $u_0 = (0, 0)$: labels $\{3, 4\}$
- $u_1 = (1/4, 0)$: labels $\{1, 4\}$
- $u_2 = (0, 1/5)$: labels $\{2, 3\}$
- u_3: intersection of $4y_1 + y_2 = 1$ and $3y_1 + 5y_2 = 1$:

From the first: $y_2 = 1 - 4y_1$. Substituting: $3y_1 + 5(1 - 4y_1) = 1 \Rightarrow 3y_1 + 5 - 20y_1 = 1 \Rightarrow -17y_1 = -4 \Rightarrow y_1 = 4/17$. Then $y_2 = 1 - 16/17 = 1/17$.

$u_3 = (4/17, 1/17)$: labels $\{1, 2\}$.

Fully labelled vertex pairs:

By the same logic as above:

Pair	$\mathcal{L}(v)$	$\mathcal{L}(u)$	Union	Fully labelled?
(v_0, u_0)	$\{1, 2\}$	$\{3, 4\}$	$\{1, 2, 3, 4\}$	Yes
(v_3, u_3)	$\{3, 4\}$	$\{1, 2\}$	$\{1, 2, 3, 4\}$	Yes

$(v_3, u_3) = ((3/19, 4/19), (4/17, 1/17))$ gives:

$$\sigma_r = \frac{(3/19, 4/19)}{7/19} = (3/7, 4/7) \tag{13.50}$$

$$\sigma_c = \frac{(4/17, 1/17)}{5/17} = (4/5, 1/5) \tag{13.51}$$

```
import nashpy as nash
import numpy as np

# Game 1
A1 = np.array([[3, -1], [2, 7]])
B1 = np.array([[-3, 1], [1, -6]])
game1 = nash.Game(A1, B1)
print("Game 1 Nash equilibria (vertex enumeration):")
for eq in game1.vertex_enumeration():
    print(" ", eq)
```

```
Game 1 Nash equilibria (vertex enumeration):
  (array([0.63636364, 0.36363636]), array([0.88888889, 0.11111111]))
```

```
# Game 2
A2 = np.array([[2, -1], [1, 3]])
B2 = np.array([[-2, 2], [1, -2]])
game2 = nash.Game(A2, B2)
print("Game 2 Nash equilibria (vertex enumeration):")
for eq in game2.vertex_enumeration():
    print(" ", eq)
```

```
Game 2 Nash equilibria (vertex enumeration):
  (array([0.42857143, 0.57142857]), array([0.8, 0.2]))
```

Solution 13.2: Solution to Exercise 2

We apply the Lemke-Howson algorithm to each game from Exercise 1.

Game 1:

$$A = \begin{pmatrix} 3 & -1 \\ 2 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} -3 & 1 \\ 1 & -6 \end{pmatrix} \tag{13.52}$$

After shifting (add 2 to A, add 7 to B):

$$A' = \begin{pmatrix} 5 & 1 \\ 4 & 9 \end{pmatrix}, \qquad B' = \begin{pmatrix} 4 & 8 \\ 8 & 1 \end{pmatrix} \tag{13.53}$$

Using known vertex structure:

From the solution to Exercise 1, the vertices and labels are:

$\mathcal{P}_r$: $v_0 = (0,0) : \{1,2\}$; $v_1 = (1/8, 0) : \{2,4\}$; $v_2 = (0, 1/8) : \{1,3\}$; $v_3 = (7/60, 1/15) : \{3,4\}$.

$\mathcal{P}_c$: $u_0 = (0,0) : \{3,4\}$; $u_1 = (1/5, 0) : \{1,4\}$; $u_2 = (0, 1/9) : \{2,3\}$; $u_3 = (8/41, 1/41) : \{1,2\}$.

Algorithm (dropping label 1):

1. Start at (v_0, u_0) with labels $\{1,2\} \cup \{3,4\} = \{1,2,3,4\}$.
2. Drop label 1 from $\mathcal{P}_r$: move from v_0 to v_1 (drops label 1, gains label 4). Now (v_1, u_0) with labels $\{2,4\} \cup \{3,4\}$. Label 4 is duplicate, so drop label 4 from $\mathcal{P}_c$.
3. Drop label 4 from $\mathcal{P}_c$: move from u_0 to u_2 (drops label 4, gains label 2). Now (v_1, u_2) with labels $\{2,4\} \cup \{2,3\}$. Label 2 is duplicate, so drop label 2 from $\mathcal{P}_r$.
4. Drop label 2 from $\mathcal{P}_r$: move from v_1 to v_3 (drops label 2, gains label 3). Now (v_3, u_2) with labels $\{3,4\} \cup \{2,3\}$. Label 3 is duplicate, so drop label 3 from $\mathcal{P}_c$.
5. Drop label 3 from $\mathcal{P}_c$: move from u_2 to u_3 (drops label 3, gains label 1). Now (v_3, u_3) with labels $\{3,4\} \cup \{1,2\} = \{1,2,3,4\}$. Fully labelled!

Normalising $v_3 = (7/60, 1/15)$ and $u_3 = (8/41, 1/41)$:

$$\sigma_r = (7/11, 4/11), \qquad \sigma_c = (8/9, 1/9) \tag{13.54}$$

Using integer pivoting:

The tableaux are:

$$\begin{aligned} T_r &= ((B')^T \;\; I \;\; \mathbf{1}) \\ &= \begin{pmatrix} 4 & 8 & 1 & 0 & 1 \\ 8 & 1 & 0 & 1 & 1 \end{pmatrix} \end{aligned} \tag{13.55}$$

$$\begin{aligned} T_c &= (I \;\; A' \;\; \mathbf{1}) \\ &= \begin{pmatrix} 1 & 0 & 5 & 1 & 1 \\ 0 & 1 & 4 & 9 & 1 \end{pmatrix} \end{aligned} \tag{13.56}$$

Non-basic variables (columns without a pivot row) in T_r are columns 1 and 2 (labels 1 and 2). Non-basic variables in T_c are columns 3 and 4 (labels 3 and 4).

Drop label 1: pivot column 1 of T_r.

Minimum ratio test: row 1 gives $1/4$; row 2 gives $1/8$. Pivot on row 2.

After pivoting row 2 (multiplying through by denominator for integer pivoting):

Row 2 becomes: $[8, 1, 0, 1, 1]$ (pivot row, divide by pivot element 8). Row 1 update: $\text{row}_1 \leftarrow 8 \cdot \text{row}_1 - 4 \cdot \text{row}_2 = [32 - 32, 64 - 4, 8 - 0, 0 - 4, 8 - 4] = [0, 60, 8, -4, 4]$.

$$T_r = \begin{pmatrix} 0 & 60 & 8 & -4 & 4 \\ 8 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{13.57}$$

Non-basic variables (columns not basic): column 2 (label 2) and column 4 (label 4). Current labels of $\mathcal{P}_r$: $\{2, 4\}$.

Duplicate label with u_0's $\{3, 4\}$: label 4. Pivot column 4 in T_c.

Minimum ratio test for T_c: row 1 gives $1/1 = 1$; row 2 gives $1/9$. Pivot on row 2.

Row 2 becomes: $[0, 1, 4, 9, 1]$. Row 1 update: $\text{row}_1 \leftarrow 9 \cdot \text{row}_1 - 1 \cdot \text{row}_2 = [9 - 0, 0 - 1, 45 - 4, 9 - 9, 9 - 1] = [9, -1, 41, 0, 8]$.

$$T_c = \begin{pmatrix} 9 & -1 & 41 & 0 & 8 \\ 0 & 1 & 4 & 9 & 1 \end{pmatrix} \tag{13.58}$$

Non-basic columns: 2 (label 2) and 3 (label 3). Labels of $\mathcal{P}_c$: $\{2, 3\}$.

Duplicate with $\mathcal{P}_r$'s $\{2, 4\}$: label 2. Pivot column 2 in T_r.

Minimum ratio test: row 1 gives $4/60 = 1/15$; row 2 gives $1/1 = 1$. Pivot on row 1.

Row 1 becomes: $[0, 60, 8, -4, 4]$. Row 2 update: $60 \cdot \text{row}_2 - 1 \cdot \text{row}_1 = [480 - 0, 60 - 60, 0 - 8, 60 + 4, 60 - 4] = [480, 0, -8, 64, 56]$.

$$T_r = \begin{pmatrix} 0 & 60 & 8 & -4 & 4 \\ 480 & 0 & -8 & 64 & 56 \end{pmatrix} \tag{13.59}$$

Non-basic columns: 3 (label 3) and 4 (label 4). Labels of $\mathcal{P}_r$: $\{3, 4\}$.

Duplicate with $\mathcal{P}_c$'s $\{2, 3\}$: label 3. Pivot column 3 in T_c.

Minimum ratio test: row 1 gives $8/41$; row 2 gives $1/4$. $8/41 \approx 0.195 < 0.25$. Pivot on row 1.

Row 1 stays: $[9, -1, 41, 0, 8]$. Row 2 update: $41 \cdot \text{row}_2 - 4 \cdot \text{row}_1 = [0 - 36, 41 + 4, 41 \cdot 4 - 4 \cdot 41, 41 \cdot 9 - 0, 41 - 32] = [-36, 45, 0, 369, 9]$.

$$T_c = \begin{pmatrix} 9 & -1 & 41 & 0 & 8 \\ -36 & 45 & 0 & 369 & 9 \end{pmatrix} \tag{13.60}$$

Non-basic columns: 1 (label 1) and 2 (label 2). Labels of $\mathcal{P}_c$: $\{1, 2\}$.

$\mathcal{P}_r$ has labels $\{3, 4\}$ and $\mathcal{P}_c$ has labels $\{1, 2\}$: union is $\{1, 2, 3, 4\}$. Fully labelled!

Reading off the solution from the basic variables:

From T_r (basic variables are columns 1 and 2 of the basis):

- Column 1 is basic in row 2: $x_1 = 56/480 = 7/60$.
- Column 2 is basic in row 1: $x_2 = 4/60 = 1/15$.

So $x = (7/60, 1/15)$, normalised: $\sigma_r = (7/11, 4/11)$.

From T_c (basic variables are columns 3 and 4):

- Column 3 basic in row 1: $y_1 = 8/41$.

- Column 4 basic in row 2: $y_2 = 9/369 = 1/41$.

So $y = (8/41, 1/41)$, normalised: $\sigma_c = (8/9, 1/9)$.

This matches the vertex enumeration result.

Game 2:

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} -2 & 2 \\ 1 & -2 \end{pmatrix} \tag{13.61}$$

After shifting: $A' = \begin{pmatrix} 4 & 1 \\ 3 & 5 \end{pmatrix}$, $B' = \begin{pmatrix} 1 & 5 \\ 4 & 1 \end{pmatrix}$.

Using known vertex structure:

$\mathcal{P}_r$: $v_0 = (0,0) : \{1,2\}$; $v_1 = (1/5, 0) : \{2,4\}$; $v_2 = (0, 1/4) : \{1,3\}$; $v_3 = (3/19, 4/19) : \{3,4\}$.

$\mathcal{P}_c$: $u_0 = (0,0) : \{3,4\}$; $u_1 = (1/4, 0) : \{1,4\}$; $u_2 = (0, 1/5) : \{2,3\}$; $u_3 = (4/17, 1/17) : \{1,2\}$.

Algorithm (dropping label 1):

1. Start at (v_0, u_0).
2. Drop label 1 from $\mathcal{P}_r$: $v_0 \to v_1$ (gains label 4). (v_1, u_0): $\{2,4\} \cup \{3,4\}$. Drop label 4 from $\mathcal{P}_c$.
3. Drop label 4 from $\mathcal{P}_c$: $u_0 \to u_2$ (gains label 2). (v_1, u_2): $\{2,4\} \cup \{2,3\}$. Drop label 2 from $\mathcal{P}_r$.
4. Drop label 2 from $\mathcal{P}_r$: $v_1 \to v_3$ (gains label 3). (v_3, u_2): $\{3,4\} \cup \{2,3\}$. Drop label 3 from $\mathcal{P}_c$.
5. Drop label 3 from $\mathcal{P}_c$: $u_2 \to u_3$ (gains label 1). (v_3, u_3): $\{3,4\} \cup \{1,2\}$. Fully labelled!

Normalising: $\sigma_r = (3/7, 4/7)$, $\sigma_c = (4/5, 1/5)$.

Using integer pivoting:

$$T_r = \begin{pmatrix} 1 & 4 & 1 & 0 & 1 \\ 5 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad T_c = \begin{pmatrix} 1 & 0 & 4 & 1 & 1 \\ 0 & 1 & 3 & 5 & 1 \end{pmatrix} \tag{13.62}$$

Drop label 1: pivot column 1 of T_r.

Min ratio test: row 1 gives $1/1 = 1$; row 2 gives $1/5$. Pivot on row 2.

Row 2 (the pivot row) is unchanged: $[5, 1, 0, 1, 1]$. Row 1: $5 \cdot [1, 4, 1, 0, 1] - 1 \cdot [5, 1, 0, 1, 1] = [0, 19, 5, -1, 4]$.

$$T_r = \begin{pmatrix} 0 & 19 & 5 & -1 & 4 \\ 5 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{13.63}$$

Labels of $\mathcal{P}_r$: $\{2,4\}$. Duplicate with u_0's $\{3,4\}$: label 4. Pivot column 4 in T_c.

Min ratio test: row 1 gives $1/1 = 1$; row 2 gives $1/5$. Pivot on row 2.

Row 2 (the pivot row) is unchanged: $[0, 1, 3, 5, 1]$. Row 1: $5 \cdot [1, 0, 4, 1, 1] - 1 \cdot [0, 1, 3, 5, 1] = [5, -1, 17, 0, 4]$.

$$T_c = \begin{pmatrix} 5 & -1 & 17 & 0 & 4 \\ 0 & 1 & 3 & 5 & 1 \end{pmatrix} \tag{13.64}$$

Labels of $\mathcal{P}_c$: $\{2,3\}$. Duplicate with $\{2,4\}$: label 2. Pivot column 2 in T_r.

Min ratio test: row 1 gives $4/19$; row 2 gives $1/1 = 1$. Pivot on row 1.

Row 1 (the pivot row) is unchanged: $[0, 19, 5, -1, 4]$. Row 2: $19 \cdot [5, 1, 0, 1, 1] - 1 \cdot [0, 19, 5, -1, 4] = [95, 0, -5, 20, 15]$.

$$T_r = \begin{pmatrix} 0 & 19 & 5 & -1 & 4 \\ 95 & 0 & -5 & 20 & 15 \end{pmatrix} \tag{13.65}$$

Labels of $\mathcal{P}_r$: $\{3, 4\}$. Duplicate with $\{2, 3\}$: label 3. Pivot column 3 in T_c.

Min ratio test: row 1 gives $4/17$; row 2 gives $1/3$. $4/17 \approx 0.235 < 0.333$. Pivot on row 1.

Row 1 (the pivot row) is unchanged: $[5, -1, 17, 0, 4]$. Row 2: $17 \cdot [0, 1, 3, 5, 1] - 3 \cdot [5, -1, 17, 0, 4] = [-15, 20, 0, 85, 5]$.

$$T_c = \begin{pmatrix} 5 & -1 & 17 & 0 & 4 \\ -15 & 20 & 0 & 85 & 5 \end{pmatrix} \tag{13.66}$$

Labels of $\mathcal{P}_c$: $\{1, 2\}$. $\mathcal{P}_r$ has $\{3, 4\}$: union is $\{1, 2, 3, 4\}$. Fully labelled!

Reading off the solution from the basic variables. In T_r, column 1 is basic in row 2 and column 2 is basic in row 1:

$$x_1 = \frac{15}{95} = \frac{3}{19}, \qquad x_2 = \frac{4}{19}, \tag{13.67}$$

so $x = (3/19, 4/19)$, normalised: $\sigma_r = (3/7, 4/7)$. In T_c, column 3 is basic in row 1 and column 4 is basic in row 2:

$$y_1 = \frac{4}{17}, \qquad y_2 = \frac{5}{85} = \frac{1}{17}, \tag{13.68}$$

so $y = (4/17, 1/17)$, normalised: $\sigma_c = (4/5, 1/5)$. This matches the vertex enumeration result.

```
import nashpy as nash
import numpy as np

# Game 1
A1 = np.array([[3, -1], [2, 7]])
B1 = np.array([[-3, 1], [1, -6]])
game1 = nash.Game(A1, B1)
print("Game 1:")
print("Vertex enumeration:")
for eq in game1.vertex_enumeration():
    print(" ", [np.round(s, 4) for s in eq])
print("Lemke-Howson (label 0):")
print(" ", [np.round(s, 4) for s in
game1.lemke_howson(initial_dropped_label=0)])
```

```
Game 1:
Vertex enumeration:
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
Lemke-Howson (label 0):
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
```

```
# Game 2
A2 = np.array([[2, -1], [1, 3]])
B2 = np.array([[-2, 2], [1, -2]])
game2 = nash.Game(A2, B2)
```

```
print("Game 2:")
print("Vertex enumeration:")
for eq in game2.vertex_enumeration():
    print(" ", [np.round(s, 4) for s in eq])
print("Lemke-Howson (label 0):")
print(" ", [np.round(s, 4) for s in
game2.lemke_howson(initial_dropped_label=0)])
```

```
Game 2:
Vertex enumeration:
```

```
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
Lemke-Howson (label 0):
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
```

```
print("All Lemke-Howson paths for Game 1:")
for eq in game1.lemke_howson_enumeration():
    print(" ", [np.round(s, 4) for s in eq])
print("All Lemke-Howson paths for Game 2:")
for eq in game2.lemke_howson_enumeration():
    print(" ", [np.round(s, 4) for s in eq])
```

```
All Lemke-Howson paths for Game 1:
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
  [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
All Lemke-Howson paths for Game 2:
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
```

```
  [array([0.4286, 0.5714]), array([0.8, 0.2])]
```

Solution 13.3: Solution to Exercise 3

The game is:

$$M_r = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 4 & 1 \\ 1 & 3 & 0 \end{pmatrix}, \qquad M_c = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 2 & 4 \\ 5 & 1 & 3 \end{pmatrix} \tag{13.69}$$

Both matrices have all positive entries so no shifting is required.

1. **Lemke-Howson algorithm.**

The tableaux are:

$$T_r = (M_c^T \;\; I_3 \;\; \mathbf{1}) = \begin{pmatrix} 2 & 1 & 5 & 1 & 0 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 & 0 & 1 \\ 1 & 4 & 3 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{13.70}$$

$$T_c = (I_3 \;\; M_r \;\; \mathbf{1}) = \begin{pmatrix} 1 & 0 & 0 & 3 & 1 & 2 & 1 \\ 0 & 1 & 0 & 2 & 4 & 1 & 1 \\ 0 & 0 & 1 & 1 & 3 & 0 & 1 \end{pmatrix} \tag{13.71}$$

We drop label 1 (pivot column 1 in T_r). Minimum ratio test:

- Row 1: $1/2$
- Row 2: $1/3$
- Row 3: $1/1 = 1$

The minimum is $1/3$ (row 2). Pivoting on row 2:

- Row 2 (the pivot row) is unchanged: $[3, 2, 1, 0, 1, 0, 1]$.
- $\text{row}_1 \leftarrow 3 \cdot \text{row}_1 - 2 \cdot \text{row}_2 = [0, -1, 13, 3, -2, 0, 1]$.
- $\text{row}_3 \leftarrow 3 \cdot \text{row}_3 - 1 \cdot \text{row}_2 = [0, 10, 8, 0, -1, 3, 2]$.

$$T_r = \begin{pmatrix} 0 & -1 & 13 & 3 & -2 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 & 0 & 1 \\ 0 & 10 & 8 & 0 & -1 & 3 & 2 \end{pmatrix} \tag{13.72}$$

The non-basic columns are 2, 3, and 5, so the labels of $\mathcal{P}_r$ are $\{2, 3, 5\}$. The duplicate with u_0's $\{4, 5, 6\}$ is label 5. Pivot column 5 in T_c.

Minimum ratio test in T_c:

- Row 1: $1/1 = 1$
- Row 2: $1/4$
- Row 3: $1/3$

The minimum is $1/4$ (row 2). Pivoting on row 2:

- Row 2 (the pivot row) is unchanged: $[0, 1, 0, 2, 4, 1, 1]$.
- $\text{row}_1 \leftarrow 4 \cdot \text{row}_1 - 1 \cdot \text{row}_2 = [4, -1, 0, 10, 0, 7, 3]$.
- $\text{row}_3 \leftarrow 4 \cdot \text{row}_3 - 3 \cdot \text{row}_2 = [0, -3, 4, -2, 0, -3, 1]$.

$$T_c = \begin{pmatrix} 4 & -1 & 0 & 10 & 0 & 7 & 3 \\ 0 & 1 & 0 & 2 & 4 & 1 & 1 \\ 0 & -3 & 4 & -2 & 0 & -3 & 1 \end{pmatrix} \tag{13.73}$$

The labels of $\mathcal{P}_c$ are $\{2, 4, 6\}$. The duplicate with $\{2, 3, 5\}$ is label 2. Pivot column 2 in T_r.

Minimum ratio test (skipping non-positive entries):

- Row 2: $1/2$
- Row 3: $2/10 = 1/5$

The minimum is $1/5$ (row 3). Pivoting on row 3:

- Row 3 (the pivot row) is unchanged: $[0, 10, 8, 0, -1, 3, 2]$.
- $\text{row}_1 \leftarrow 10 \cdot \text{row}_1 + 1 \cdot \text{row}_3 = [0, 0, 138, 30, -21, 3, 12]$.

- $\text{row}_2 \leftarrow 10 \cdot \text{row}_2 - 2 \cdot \text{row}_3 = [30, 0, -6, 0, 12, -6, 6]$.

$$T_r = \begin{pmatrix} 0 & 0 & 138 & 30 & -21 & 3 & 12 \\ 30 & 0 & -6 & 0 & 12 & -6 & 6 \\ 0 & 10 & 8 & 0 & -1 & 3 & 2 \end{pmatrix} \tag{13.74}$$

The labels of $\mathcal{P}_r$ are $\{3, 5, 6\}$. The duplicate with $\{2, 4, 6\}$ is label 6. Pivot column 6 in T_c.

Minimum ratio test (skipping non-positive entries):

- Row 1: $3/7$
- Row 2: $1/1 = 1$

The minimum is $3/7$ (row 1). Pivoting on row 1:

- Row 1 (the pivot row) is unchanged: $[4, -1, 0, 10, 0, 7, 3]$.
- $\text{row}_2 \leftarrow 7 \cdot \text{row}_2 - 1 \cdot \text{row}_1 = [-4, 8, 0, 4, 28, 0, 4]$.
- $\text{row}_3 \leftarrow 7 \cdot \text{row}_3 + 3 \cdot \text{row}_1 = [12, -24, 28, 16, 0, 0, 16]$.

$$T_c = \begin{pmatrix} 4 & -1 & 0 & 10 & 0 & 7 & 3 \\ -4 & 8 & 0 & 4 & 28 & 0 & 4 \\ 12 & -24 & 28 & 16 & 0 & 0 & 16 \end{pmatrix} \tag{13.75}$$

The labels of $\mathcal{P}_c$ are $\{1, 2, 4\}$. Together with $\mathcal{P}_r$'s $\{3, 5, 6\}$ the union is $\{1, 2, 3, 4, 5, 6\}$: fully labelled.

Reading off the basic variables. From T_r, column 1 is basic in row 2 and column 2 in row 3:

$$x_1 = \frac{6}{30} = \frac{1}{5}, \qquad x_2 = \frac{2}{10} = \frac{1}{5}, \qquad x_3 = 0, \tag{13.76}$$

so $\sigma_r = (1/2, 1/2, 0)$. From T_c, column 5 is basic in row 2 and column 6 in row 1:

$$y_1 = 0, \qquad y_2 = \frac{4}{28} = \frac{1}{7}, \qquad y_3 = \frac{3}{7}, \tag{13.77}$$

so $\sigma_c = (0, 1/4, 3/4)$. We can confirm this with nashpy.

```
import nashpy as nash
import numpy as np

M_r = np.array([[3, 1, 2], [2, 4, 1], [1, 3, 0]])
M_c = np.array([[2, 3, 1], [1, 2, 4], [5, 1, 3]])
game = nash.Game(M_r, M_c)

print("Vertex enumeration (all Nash equilibria):")
for eq in game.vertex_enumeration():
    print(" ", [np.round(s, 4) for s in eq])

print("\nLemke-Howson from each starting label:")
for label in range(6):
    eq = game.lemke_howson(initial_dropped_label=label)
    print(f"  Label {label}: {[np.round(s, 4) for s in eq]}")
```

```
Vertex enumeration (all Nash equilibria):
```

```
  [array([0.5, 0.5, 0. ]), array([-0.  ,  0.25,  0.75])]

Lemke-Howson from each starting label:
  Label 0: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
  Label 1: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
  Label 2: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
  Label 3: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
  Label 4: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
  Label 5: [array([0.5, 0.5, 0. ]), array([0.  , 0.25, 0.75])]
```

The computation confirms that different choices of dropped label can lead to different Nash equilibria being found (when multiple exist) or trace different paths to the same equilibrium. The key insight is that the algorithm always terminates at a fully labelled vertex pair, but the path taken depends on the initial dropped label.

2. **Business strategy interpretation.**

The algorithm returns $\sigma_r = (1/2, 1/2, 0)$ and $\sigma_c = (0, 1/4, 3/4)$. Café A splits evenly between **Price** and **Quality** and never **Advertises**; Café B never competes on **Price**, playing **Quality** a quarter of the time and **Advertising** the rest.

- Café A is indifferent between Price and Quality against σ_c: both earn an expected payoff of $7/4$, while Advertising earns only $3/4$ and is dropped. Mixing equally between its two best responses is what keeps Café B indifferent in turn.
- Café B avoids Price because, against an opponent who never advertises, Price is the weaker option: both Quality and Advertising earn $5/2$ against σ_r, whereas Price earns only $3/2$.

The equilibrium is mixed, meaning neither café can improve its expected payoff by deviating unilaterally. This reflects a market where no single strategy dominates across all customer profiles; each café must remain unpredictable across the strategies it does use to prevent the rival from exploiting a fixed choice.

Solution 13.4: Solution to Exercise 4

Theorem: Under the assumptions of nondegeneracy and the property that dropping a label from any fully labelled vertex pair leads to a distinct fully labelled vertex pair, the number of Nash equilibria is odd.

Proof:

Each vertex of $\mathcal{P}_r$ carries exactly m labels and each vertex of $\mathcal{P}_c$ exactly n, so a vertex pair (x, y) carries $m + n$ labels counted with multiplicity. Call the pair **completely labelled** when these labels are all distinct, so that every label in $\{1, \ldots, m+n\}$ appears exactly once; this is the fully labelled condition. By nondegeneracy the completely labelled pairs are exactly the Nash equilibria together with the artificial pair $(0, 0)$, whose labels are $\{1, \ldots, m\}$ (from $x = 0$) and $\{m+1, \ldots, m+n\}$ (from $y = 0$).

Fix a label $k \in \{1, \ldots, m+n\}$ and call a pair ***k*-almost completely labelled** if every label except possibly k appears. Under nondegeneracy such a pair is either completely labelled, or carries every label except k with exactly one other label duplicated.

Form the graph H_k whose nodes are the k-almost completely labelled pairs, with an edge joining two pairs that differ by a single Lemke-Howson pivot: dropping the duplicated label from one of the two vertices that carry it. The degree of a node is then determined by two cases:

- A node that is not completely labelled has exactly one duplicated label, carried by a vertex of $\mathcal{P}_r$ and a vertex of $\mathcal{P}_c$. Dropping it on either side gives a pivot, so the node has **degree 2**.
- A completely labelled node has no duplicated label, so the only available move is to drop label k itself. It has **degree 1**.

Every connected component of H_k is therefore a path or a cycle, and the degree-1 nodes are precisely the completely labelled pairs. In a disjoint union of paths and cycles the degree-1 nodes come in pairs, one at each end of every path, so their total number is **even**.

The completely labelled pairs are the Nash equilibria together with the artificial pair $(0, 0)$. Hence

$$\big|\,\text{Nash equilibria}\big| + 1 \;\text{ is even}, \tag{13.78}$$

so the number of Nash equilibria is **odd**. □

```
import nashpy as nash
import numpy as np

# Demonstrate odd number of equilibria for examples
games = [
    (np.array([[3, -1], [2, 7]]), np.array([[-3, 1], [1, -6]]), "Game 1"),
    (np.array([[2, -1], [1, 3]]), np.array([[-2, 2], [1, -2]]), "Game 2"),
    (np.array([[3, 1, 2], [2, 4, 1], [1, 3, 0]]),
     np.array([[2, 3, 1], [1, 2, 4], [5, 1, 3]]), "Coffee shop"),
]

for A, B, name in games:
    game = nash.Game(A, B)
    eqs = list(game.vertex_enumeration())
    print(f"{name}: {len(eqs)} Nash equilibrium/ia (odd: {len(eqs) % 2 ==
1})")
    for eq in eqs:
        print("  ", [np.round(s, 4) for s in eq])
```

```
Game 1: 1 Nash equilibrium/ia (odd: True)
   [array([0.6364, 0.3636]), array([0.8889, 0.1111])]
```

```
Game 2: 1 Nash equilibrium/ia (odd: True)
   [array([0.4286, 0.5714]), array([0.8, 0.2])]
```

```
Coffee shop: 1 Nash equilibrium/ia (odd: True)
   [array([0.5, 0.5, 0. ]), array([-0.  ,  0.25,  0.75])]
```

14. Routing Games

Selfish agents choosing routes through a shared network rarely produce the socially optimal outcome. This chapter studies routing games and the Price of Anarchy, quantifying how much individual optimisation can degrade collective performance.

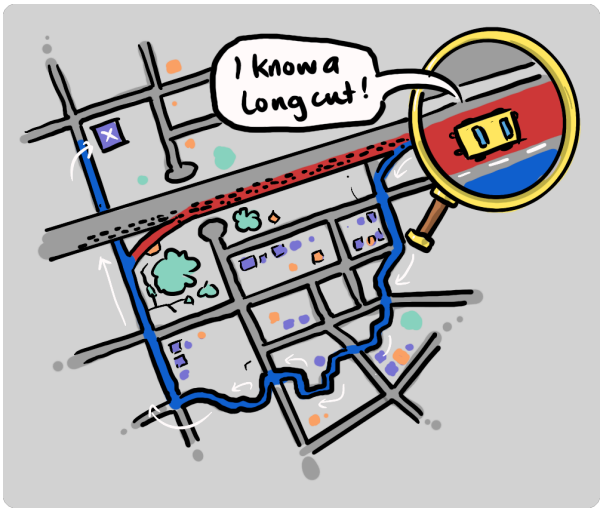


Figure 14.1: A route only stays quick for as long as few people take it. Once every driver spots the same shortcut and switches onto it, the shortcut is no quicker than the road it replaced.

14.1 Motivating Example: Competing delivery companies

In a busy city, two rival delivery companies, **QuickShip** and **TurboExpress**, operate daily distribution routes to deliver packages to the central business district.

Each company starts from its own warehouse, located in different suburbs on opposite sides of the city. Every morning, the companies dispatch their fleets toward the central depot, but must decide how to route their vans:

- Each company can take its own **dedicated service road** to the depot.
- Alternatively, both companies can use a **shared ring road** that offers a faster connection, but becomes congested as total traffic increases.

The travel costs for each route depend on the level of congestion:

- For QuickShip (source s_1):
  - Dedicated service road cost: $c_{s_1}(x) = x^2$, where x is the number of QuickShip vans using this road.
- For TurboExpress (source s_2):
  - Dedicated service road cost: $c_{s_2}(x) = \frac{3}{2}x$.
- For the shared ring road (available to both companies):
  - Shared road cost: $c_{\text{shared}}(x) = x$, where x is the total number of vans from both companies using this route.

Each source has a total demand of $r = 1/2$ units of traffic to send to the sink.

This is shown diagrammatically in Figure 14.2.

Each company decides independently how to distribute its fleet across the available routes, aiming to minimise its own delivery costs. However, since the shared road's cost depends on total usage, each company's decision affects the other's outcome.

This setting creates a **routing game**, where decentralised decisions lead to interdependent costs. We will analyse this scenario to explore the concept of **Nash equilibrium** in network routing, and investigate how self-interested routing can result in stable, but potentially inefficient, traffic patterns.

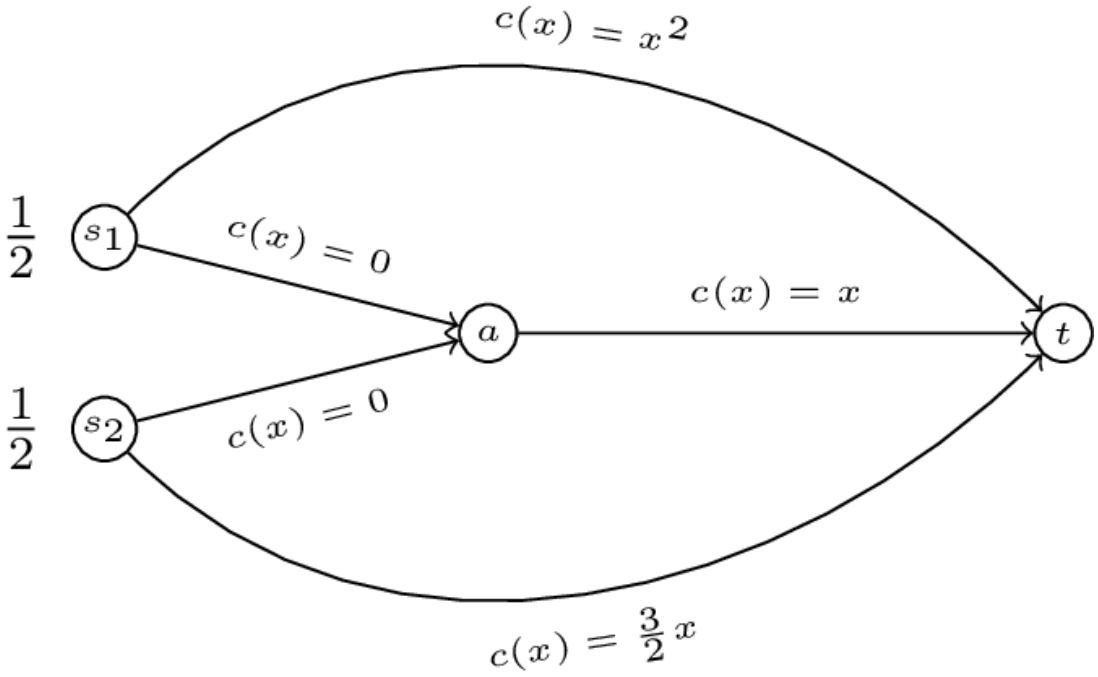


Figure 14.2: The routes available to QuickShip and TurboExpress.

14.2 Theory

14.2.1 Definition: Routing Game

A **routing game** (G, r, c) is defined on a graph $G = (V, E)$ with a defined set of sources s_i and sinks t_i. Each source-sink pair corresponds to a set of traffic (also called a commodity) r_i that must travel along the edges of G from s_i to t_i. Every edge e of G has associated to it a nonnegative, continuous and nondecreasing cost function (also called latency function) c_e.

14.2.1.1 Example: The Routing Game Representation of the Delivery Companies Game

For the routing game in Figure 14.2:

- $G = (V, E)$ with $V = \{s_1, s_2, a, t\}$ and $E = \{(s_1, t), (s_1, a), (s_2, a), (s_2, t), (a, t)\}$
- $r_1 = r_2 = 1/2$
- $c_{s_1,t}(x) = x^2, c_{s_2,t}(x) = 3x/2, c_{a,t}(x) = x, c_{s_1,a}(x) = c_{s_2,a} = 0$

14.2.2 Definition: Set of Paths

For any given (G, r, c) we denote by $\mathcal{P}_i$ the set of paths available to commodity i.

14.2.3 Definition: Feasible Flow

On a routing game we define a flow f as a vector representing the quantity of traffic along the various paths, f is a vector index by $\mathcal{P}$. We call f **feasible** if:

$$\sum_{P \in \mathcal{P}_i} f_P = r_i \tag{14.1}$$

14.2.3.1 Example: Feasible Flows for the Delivery Companies Game

For the routing game in Figure 14.2:

$$\mathcal{P}_1 = \{(s_1, a, t), (s_1, t)\} \tag{14.2}$$

and

$$\mathcal{P}_2 = \{(s_2, a, t), (s_2, t)\} \tag{14.3}$$

The set of all possible paths is denoted by $\mathcal{P} = \cup_i \mathcal{P}_i$.

14.2.4 Definition: Cost function

For any routing game (G, r, c) we define a **cost function** $C(f)$:

$$C(f) = \sum_{P \in \mathcal{P}} c_P(f_P) f_P \tag{14.4}$$

Where c_P denotes the cost function of a particular path P: $c_P(x) = \sum_{e \in P} c_e(x)$. Note that any flow f induces a flow on edges:

$$f_e = \sum_{P \in \mathcal{P} \text{ if } e \in P} f_P \tag{14.5}$$

So we can re-write the cost function as:

$$C(f) = \sum_{e \in E} c_e(f_e) f_e \tag{14.6}$$

14.2.4.1 Example: Cost function for the Delivery Companies Game

For the routing game in Figure 14.2:

Let $f = (\alpha, 1/2 - \alpha, 1/2 - \beta, \beta)$ as shown in Figure 14.3.

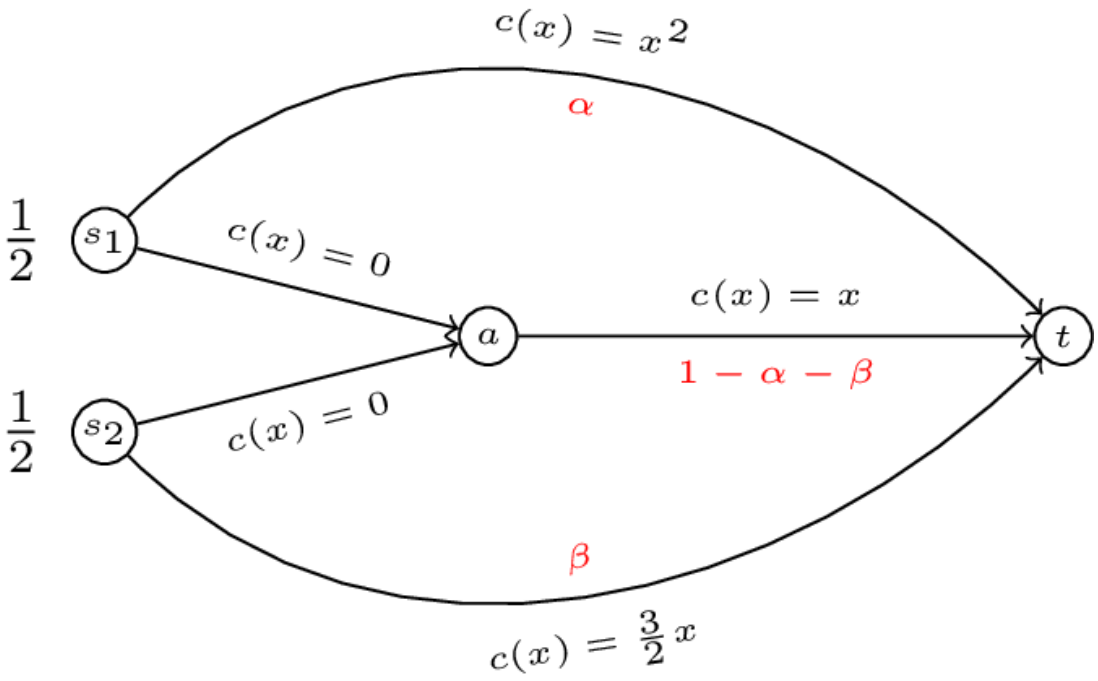


Figure 14.3: The vector f showing the flow on the game

The cost of $f(\alpha, 1/2 - \alpha, 1/2 - \beta, \beta)$ is given by:

$$
\begin{aligned}
C(f) &= \alpha^2 \times \alpha + 3/2\beta \times \beta + (1-\alpha-\beta) \times (1-\alpha-\beta) \\
&= \alpha^3 + 3/2\beta^2 + \alpha^2 + 2\alpha\beta + \beta^2 - 2\alpha - 2\beta + 1
\end{aligned} \tag{14.7}
$$

14.2.5 Definition: Optimal Flow

For a routing game (G, r, c) we define the optimal flow f^* as the solution to the following optimisation problem:

Minimise $\sum_{e \in E} c_e(f_e) f_e$:

Subject to:

$$
\begin{aligned}
\sum_{P \in \mathcal{P}_i} f_P &= r_i \qquad \text{for all } i \\
f_e &= \sum_{P \in \mathcal{P} \text{ if } e \in P} f_P \quad \text{for all } e \in E \\
f_P &\geq 0
\end{aligned} \tag{14.8}
$$

14.2.5.1 Example: Optimal Flow for the Delivery Companies Games

In Figure 14.3 this corresponds to:

Minimise $C(\alpha, \beta) = \alpha^3 + 3/2\beta^2 + \alpha^2 + 2\alpha\beta + \beta^2 - 2\alpha - 2\beta + 1$:

Subject to:

$$
\begin{aligned}
0 &\leq \alpha \\
\alpha &\leq 1/2 \\
0 &\leq \beta \\
\beta &\leq 1/2
\end{aligned} \tag{14.9}
$$

A plot of $C(\alpha, \beta)$ is shown in Figure 14.4.

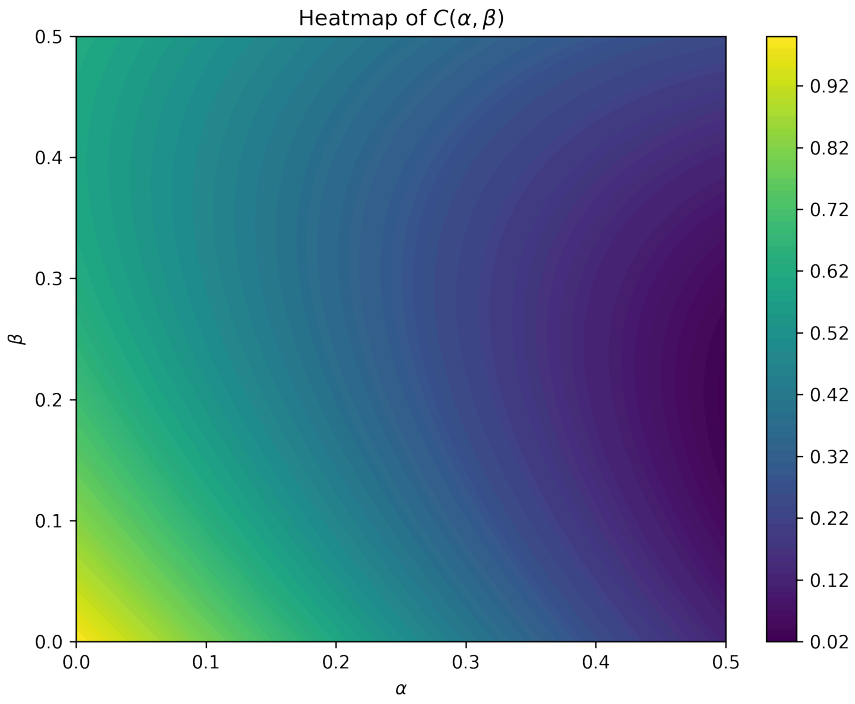


Figure 14.4: The plot of (α, β).

The minimal point looks to be near the right boundary of the square, although it is unclear if it is on the boundary or not. It is in fact **not on the boundary** taking this fact as an assumption, identity the optimal flow using the Karush-Kuhn-Tucker conditions.

Let us first define the Lagrangian:

$$\mathcal{L}(\alpha, \beta, \lambda) = C(\alpha, \beta) - \lambda_1 \alpha - \lambda_2(\alpha - 1/2) - \lambda_3 \beta - \lambda_4 \beta \tag{14.10}$$

If f^* is the minimal point then the complementary slackness implies that $\lambda_i = 0$ for all i, since this point does not sit on any boundary which would make the corresponding constraint function $g_i(f^*) = 0$.

Thus:

$$\mathcal{L}(\alpha, \beta, \lambda) = C(\alpha, \beta) \tag{14.11}$$

We note that the feasiblity constraints all hold for f^* so we are left to check stationarity:

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{\partial C}{\partial \alpha} = 3\alpha^2 - 2(1 - \alpha - \beta) = 0 \tag{14.12}$$

and

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{\partial C}{\partial \beta} = 3\beta - 2(1 - \alpha - \beta) = 0 \tag{14.13}$$

which gives:

$$\beta = \frac{2(1 - \alpha)}{5} \tag{14.14}$$

Substituting this in to the first equation gives:

$$3\alpha^2 - 6/5(1 - \alpha) = 0 \tag{14.15}$$

which has solutions:

$$\left\{ -\frac{1}{5} - \frac{\sqrt{11}}{5}, \frac{\sqrt{11}}{5} - \frac{1}{5} \right\} \tag{14.16}$$

Only one of those is positive satisfying the constraints, substituting it in to (14.14) gives:

$$\beta = \frac{12}{25} - \frac{2\sqrt{11}}{25} \tag{14.17}$$

thus the optimal flow is:

$$f^* = \left(\frac{\sqrt{11}}{5} - \frac{1}{5}, \frac{12}{25} - \frac{2\sqrt{11}}{25} \right) \tag{14.18}$$

If we take a closer look at f^* in our example, we see that traffic from the first commodity travels across two paths: (s_1, t) and (s_1, a, t). The cost along the first path is given by:

$$C_{s_1,t}(f^*) = \frac{12}{25} - \frac{2\sqrt{11}}{25} \approx 0.2146 \tag{14.19}$$

The cost along the second path is given by:

$$C_{s_1,a,t}(f^*) = \frac{18}{25} - \frac{3\sqrt{11}}{25} \approx 0.3220 \tag{14.20}$$

Thus traffic going along the second path is experiencing a higher cost. In practice, they would deviate some of their traffic from the second path to the first.

This leads to the definition of a Nash flow.

14.2.6 Definition: Nash Flow

For a routing game (G, r, c) a flow $\tilde{f}$ is called a **Nash flow** if and only if for every commodity i and any two paths $P_1, P_2 \in \mathcal{P}_i$ such that $f_{P_1} > 0$ then:

$$c_{P_1}(f) \leq c_{P_2}(f) \tag{14.21}$$

A Nash flow ensures that all used paths have minimal costs.

14.2.6.1 Example: Nash Flow for Delivery companies game

For Figure 14.3 if we assume that both commodities use all paths available to them:

$$\begin{aligned} \alpha^2 &= 1 - \alpha - \beta \\ \frac{3\beta}{2} &= 1 - \alpha - \beta \end{aligned} \tag{14.22}$$

Solving this gives: $\beta = \frac{2}{5}(1-\alpha) \Rightarrow x^2 + \frac{3}{5}x - \frac{3}{5}$ this gives

$$\left\{ -\frac{3}{10} - \frac{\sqrt{69}}{10}, -\frac{3}{10} + \frac{\sqrt{69}}{10} \right\} \tag{14.23}$$

Neither of these two values are within our region thus there is no Nash flow where all paths are used.

Let us assume that $\alpha = 1/2$ (i.e. commodity 1 does not use $P_{s_1,a,t}$). Assuming that the second commodity uses both available paths we have:

$$\frac{3}{2}\beta = \frac{1}{2} - \beta \Rightarrow \beta = \frac{1}{5} \tag{14.24}$$

Thus we have $\tilde{f} = (1/2, 1/5)$ which gives a cost of $C\left(\tilde{f}\right) = 11/40$ which is higher than the optimal cost!

What if we had assumed that $\beta = 1/2$?

This would have given $\alpha^2 = 1/2 - \alpha$ which has solutions:

$$\left\{ -\frac{1}{2} + \frac{\sqrt{3}}{2}, -\frac{\sqrt{3}}{2} - \frac{1}{2} \right\} \tag{14.25}$$

Taking the first value for α which does lie in the feasible region. The cost of the path $P_{s_2,a,t}$ is then approximately 0.134 however the cost of the path $P_{s_2,t}$ is approximately 0.75 thus the second commodity should deviate. We can carry out these same checks with all other possibilities to verify that $\tilde{f} = (1/2, 1/5)$ is a Nash flow.

14.2.7 Definition: Potential Function

Given a routing game (G, r, c) we define the **potential function** of a flow as:

$$\Phi(f) = \sum_{e \in E} \int_0^{f_e} c_e(x)dx \tag{14.26}$$

14.2.7.1 Example: Potential Function for the delivery company game

The potential function for Figure 14.3 is given by:

$$\begin{aligned}\Phi((\alpha,\beta)) &= \int_0^{\alpha} x^2 dx + \int_0^{1-\alpha-\beta} x dx + \int_0^{\beta} 3/2 x dx \\ &= \frac{\alpha^3}{3} + \frac{3\beta^2}{4} + \frac{(1-\alpha-\beta)^2}{2}\end{aligned} \tag{14.27}$$

14.2.8 Theorem: Nash Flow minimises the potential function

A feasible flow $\tilde{f}$ is a Nash flow for the routing game (G, r, c) if and only if it is a minima for $\Phi(f)$.

We state this result without proof. It follows from the first-order (KKT) optimality conditions for minimising the potential Φ, which for the non-atomic routing game coincide with the equal-cost condition defining a Nash flow. The KKT conditions are the tool used to check particular flows in the exercises below.

14.2.9 Definition: Marginal Cost

If c is a differentiable cost function then we define the **marginal cost** function c^* as:

$$c^* = \frac{d}{dx}(xc(x)) \tag{14.28}$$

14.2.9.1 Example: The marginal costs for the delivery company game

The marginal costs for Figure 14.3 are given by:

$c^*_{s_1,t}(x) = 3x^2, c_{s_2,t}(x) = 3x, c_{a,t}(x) = 2x, c_{s_1,a}(x) = c_{s_2,a} = 0$

The corresponding routing game (G, r, c^*) is shown in Figure 14.5.

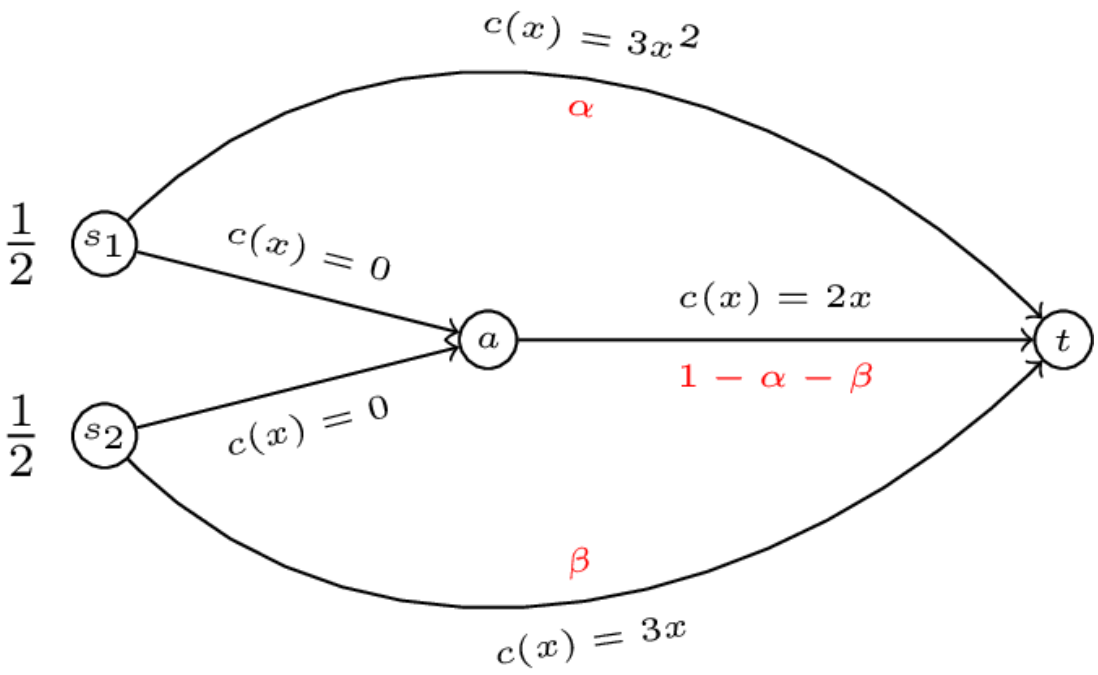


Figure 14.5: The routes available to QuickShip and TurboExpress with marginal costs instead of costs.

14.2.10 Theorem: Optimal Flow is a Nash Flow for Marginal Costs

A feasible flow f^* is an optimal flow for (G, r, c) if and only if f^* is a Nash flow for (G, r, c^*).

We state this result without proof. It follows on comparing the optimality conditions for the two problems: the optimal flow minimises the total cost $C(f) = \sum_e f_e c_e(f_e)$, whose first-order condi-

tions equate the marginal costs $c_e^* = \frac{d}{dx}(xc_e(x))$ across used paths, and these are exactly the equal-cost conditions for a Nash flow of the game with costs c^*.

14.3 Exercises

Exercise 14.1:

Determine the optimal and Nash flows for the routing games shown in Figure 14.6 and Figure 14.7.

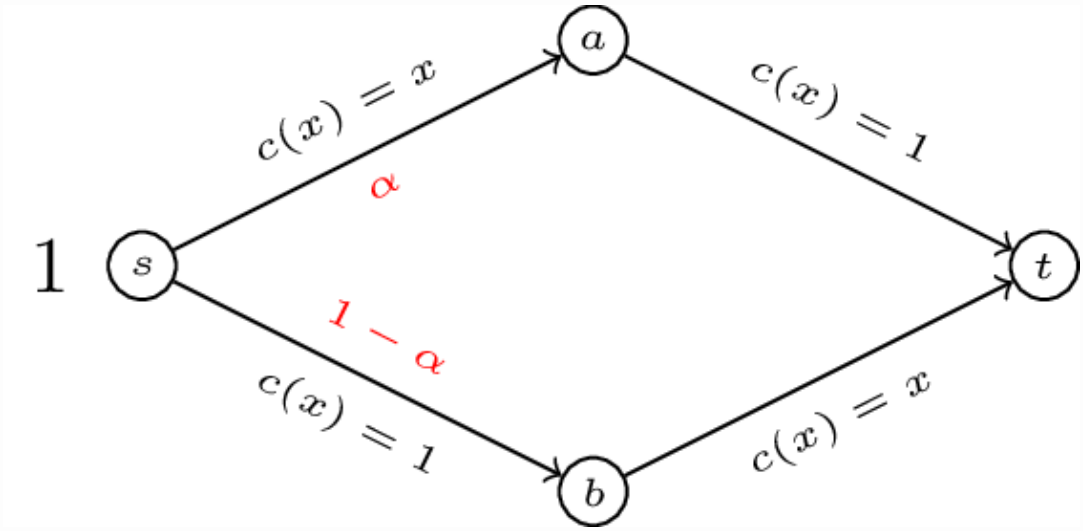


Figure 14.6: A routing game forming the basis of Braess's paradox

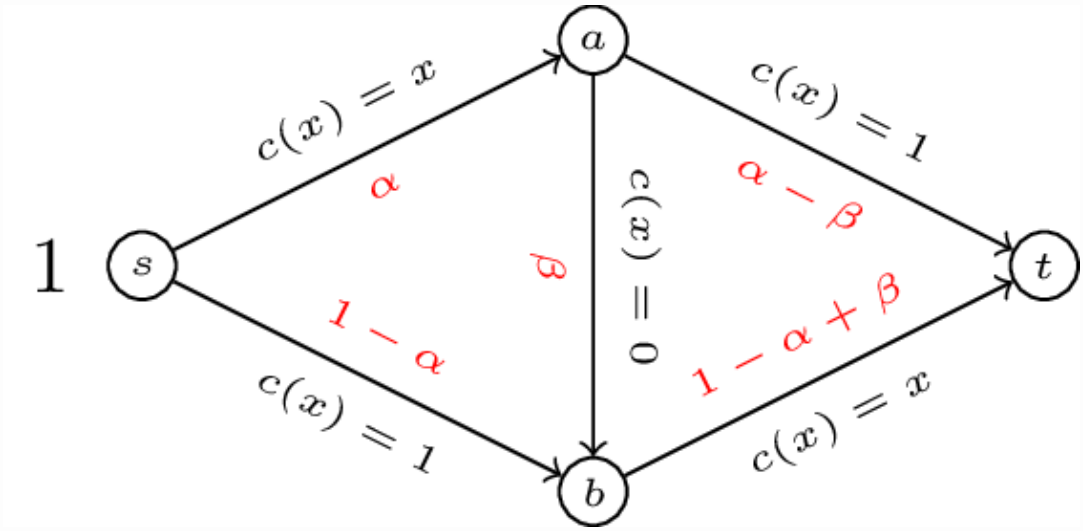


Figure 14.7: A routing game known as Braess's paradox.

Exercise 14.2:

Use Theorem: Nash Flow minimises the potential function and Appendix: Karush-Kuhn-Tucker Conditions to verify that $\tilde{f} = (1/2, 1/5)$ is a Nash flow for Figure 14.3.

Exercise 14.3:

Use Theorem: Optimal Flow is a Nash Flow for Marginal Costs to confirm that

$$f = \left(\frac{\sqrt{11}}{5} - \frac{1}{5}, \frac{12}{25} - \frac{2\sqrt{11}}{25} \right) \tag{14.29}$$

is a Nash flow for Figure 14.3.

14.4 Programming

Scipy has functionality for optimising function within certain boundaries, thanks to Theorem: Nash Flow minimises the potential function this can be used to find not only optimal flows but also Nash flows.

In Appendix: Karush-Kuhn-Tucker Conditions is an example showing how to use Scipy to optimise a function. Here is another one:

```
import numpy as np
import scipy.optimize

def objective(f):
    alpha, beta = f
    return alpha ** 3 / 3 + 3 * beta ** 2 / 4 + (1 - alpha - beta) ** 2 / 2

def g_1(f):
    return f[0]

def g_2(f):
    return f[1]

def g_3(f):
    return 1 / 2 - f[0]

def g_4(f):
    return 1 / 2 - f[1]

constraints = [
    {'type': 'ineq', 'fun': g_1},
    {'type': 'ineq', 'fun': g_2},
    {'type': 'ineq', 'fun': g_3},
    {'type': 'ineq', 'fun': g_4},
]

res = scipy.optimize.minimize(objective, [0, 0], constraints=constraints)
print(res)
```

```
     message: Optimization terminated successfully
     success: True
      status: 0
         fun: 0.11666666666666675
           x: [ 5.000e-01  2.000e-01]
         nit: 4
         jac: [-5.000e-02 -9.313e-10]
```

```
        nfev: 12
        njev: 4
 multipliers: [ 0.000e+00  0.000e+00  5.000e-02  0.000e+00]
```

14.4.1 Notable Research

The original paper introducing the notion of Nash flows is (Wardrop, 1952). An important phenomenon in this field is Braess's Paradox, first described in (Braess, 1968), which shows that adding capacity to a network can paradoxically increase overall congestion. This effect is not purely theoretical and has been observed in real-world settings (Steinberg & Zangwill, 1983; Youn et al., 2008).

The inefficiencies that arise in congestion networks due to self-interested behaviour have led to the concept of the **Price of Anarchy**, defined as $C\left(\tilde{f}\right)/C(f^*)$. This concept was introduced in (Roughgarden & Tardos, 2002). Various applications have been explored, including (Knight & Harper, 2013), which studied the inefficiencies caused by patients selecting hospitals based on individual preferences.

14.4.2 Conclusion

Table 14.1 summarises the key concepts.

Concept	Definition
Feasible Flow	Any flow satisfying demand and non-negativity constraints
Optimal Flow (f^*)	Flow minimizing the total system cost $C(f)$
Nash Flow ($\tilde{f}$)	Flow where no player can reduce their cost by unilaterally switching routes
Potential Function	$\Phi(f) = \sum_{e \in E} \int_0^{f_e} c_e(x)dx$
Marginal Cost	$c^*(x) = \frac{d}{dx}(xc(x))$

Table 14.1: Summary of routing games

Attention

In routing games, individual rationality does not guarantee system-wide efficiency. Tools like potential functions and marginal costs provide powerful methods to identify Nash and Optimal flows in decentralized network systems.

14.5 Solutions

Solution 14.1: Solution to Exercise 1

Network without the super road

The network in Figure 14.6 has a single source s, a single sink t, two intermediate nodes a and b, and a total demand of $r = 1$. The edges and their cost functions are:

$$c_{s,a}(x) = x, \quad c_{a,t}(x) = 1, \quad c_{s,b}(x) = 1, \quad c_{b,t}(x) = x \tag{14.30}$$

Let α denote the flow on path $P_1 = (s, a, t)$ and $1 - \alpha$ the flow on path $P_2 = (s, b, t)$, with $0 \leq \alpha \leq 1$.

Optimal flow (without super road)

The cost function is:

$$C(\alpha) = c_{s,a}(\alpha) \cdot \alpha + c_{a,t}(\alpha) \cdot \alpha + c_{s,b}(1-\alpha) \cdot (1-\alpha) + c_{b,t}(1-\alpha) \cdot (1-\alpha) \tag{14.31}$$

$$C(\alpha) = \alpha^2 + \alpha + (1-\alpha) + (1-\alpha)^2 = \alpha^2 + (1-\alpha)^2 + 1 \tag{14.32}$$

To minimise, take the derivative and set it to zero:

$$\frac{dC}{d\alpha} = 2\alpha - 2(1-\alpha) = 4\alpha - 2 = 0 \Longrightarrow \alpha^* = \frac{1}{2} \tag{14.33}$$

The optimal flow is $f^* = \alpha^* = 1/2$, giving:

$$C(f^*) = \frac{1}{4} + \frac{1}{4} + 1 = \frac{3}{2} \tag{14.34}$$

Nash flow (without super road)

A Nash flow requires the cost along every used path to be equal. The cost along each path at flow $(\alpha, 1-\alpha)$ is:

$$c_{P_1}(\alpha) = \alpha + 1, \qquad c_{P_2}(1-\alpha) = 1 + (1-\alpha) = 2 - \alpha \tag{14.35}$$

Setting $c_{P_1} = c_{P_2}$:

$$\alpha + 1 = 2 - \alpha \Longrightarrow \tilde{\alpha} = \frac{1}{2} \tag{14.36}$$

So the Nash flow is $\tilde{f} = 1/2$, giving $\tilde{f} = f^*$ and:

$$C\left(\tilde{f}\right) = \frac{3}{2} \tag{14.37}$$

In this case the Nash flow coincides with the optimal flow, so the **Price of Anarchy** is 1.

```
import numpy as np
import scipy.optimize

def cost_without_super_road(f):
    alpha = f[0]
    return alpha ** 2 + (1 - alpha) ** 2 + 1

def potential_without_super_road(f):
    alpha = f[0]
    # Phi = int_0^alpha x dx + int_0^alpha 1 dx + int_0^(1-alpha) 1 dx +
int_0^(1-alpha) x dx
    return alpha ** 2 / 2 + alpha + (1 - alpha) + (1 - alpha) ** 2 / 2

constraints = [
    {'type': 'ineq', 'fun': lambda f: f[0]},
    {'type': 'ineq', 'fun': lambda f: 1 - f[0]},
]

res_opt = scipy.optimize.minimize(cost_without_super_road, [0.5],
constraints=constraints)
res_nash = scipy.optimize.minimize(potential_without_super_road, [0.5],
constraints=constraints)

print(f"Optimal flow alpha = {res_opt.x[0]:.4f}, cost = {res_opt.fun:.4f}")
print(f"Nash flow alpha = {res_nash.x[0]:.4f}, cost =
{cost_without_super_road(res_nash.x):.4f}")
```

```
Optimal flow alpha = 0.5000, cost = 1.5000
Nash flow alpha = 0.5000, cost = 1.5000
```

Network with the super road (Braess's Paradox)

The network in Figure 14.7 adds the edge $a \to b$ with $c_{a,b}(x) = 0$. Using the flow vector shown in the figure, let α be the total flow entering node a from s, and β be the flow on the edge $a \to b$. The three paths are:

- $P_1 = (s, a, t)$: flow $\alpha - \beta$
- $P_2 = (s, b, t)$: flow $1 - \alpha + \beta$
- $P_3 = (s, a, b, t)$: flow β

with constraints $0 \le \beta \le \alpha$, $0 \le \alpha \le 1$, $0 \le 1 - \alpha + \beta \le 1$.

The edge flows are: $f_{s,a} = \alpha$, $f_{a,t} = \alpha - \beta$, $f_{s,b} = 1 - \alpha$, $f_{b,t} = 1 - \alpha + \beta$, $f_{a,b} = \beta$.

The cost function is:

$$\begin{aligned} C(\alpha, \beta) &= c_{s,a}(\alpha) \cdot \alpha + c_{a,t}(\alpha - \beta) \cdot (\alpha - \beta) \\ &\quad + c_{s,b}(1 - \alpha) \cdot (1 - \alpha) + c_{b,t}(1 - \alpha + \beta) \cdot (1 - \alpha + \beta) \\ &\quad + c_{a,b}(\beta) \cdot \beta \\ &= \alpha^2 + (\alpha - \beta) + 1 \cdot (1 - \alpha) + (1 - \alpha + \beta)^2 + 0 \end{aligned} \tag{14.38}$$

Optimal flow (with super road)

The potential function is:

$$\Phi(\alpha, \beta) = \frac{\alpha^2}{2} + (\alpha - \beta) + \frac{(\alpha - \beta)^2}{2} \cdot 0 + (1 - \alpha) + \frac{(1 - \alpha + \beta)^2}{2} + 0 \tag{14.39}$$

More carefully, since $c_{s,a}(x) = x$, $c_{a,t}(x) = 1$, $c_{s,b}(x) = 1$, $c_{b,t}(x) = x$, $c_{a,b}(x) = 0$:

$$\Phi(\alpha, \beta) = \frac{\alpha^2}{2} + (\alpha - \beta) \cdot 1 + (1 - \alpha) \cdot 1 + \frac{(1 - \alpha + \beta)^2}{2} \tag{14.40}$$

Taking partial derivatives:

$$\frac{\partial C}{\partial \alpha} = 2\alpha + 1 - 1 - 2(1 - \alpha + \beta)(-1) = 2\alpha + 2(1 - \alpha + \beta) - 0 + 1 - 1 \tag{14.41}$$

It is cleaner to work directly with the cost:

$$C(\alpha, \beta) = \alpha^2 + \alpha - \beta + (1 - \alpha) + (1 - \alpha + \beta)^2 \tag{14.42}$$

$$\frac{\partial C}{\partial \alpha} = 2\alpha - 1 + 1 - 2(1 - \alpha + \beta) = 2\alpha - 2(1 - \alpha + \beta) = 4\alpha + 2\beta - 2 \tag{14.43}$$

$$\frac{\partial C}{\partial \beta} = -1 + 2(1 - \alpha + \beta) = 2\beta - 2\alpha + 1 \tag{14.44}$$

Setting both to zero:

$$4\alpha + 2\beta = 2 \implies 2\alpha + \beta = 1 \tag{14.45}$$

$$2\beta - 2\alpha + 1 = 0 \implies \beta = \alpha - \frac{1}{2} \tag{14.46}$$

Substituting: $2\alpha + \alpha - 1/2 = 1 \implies 3\alpha = 3/2 \implies \alpha^* = 1/2$, $\beta^* = 0$.

The optimal flow is $f^* = (\alpha^*, \beta^*) = (1/2, 0)$, which is the same as the network without the super road, with cost:

$$C(f^*) = \frac{1}{4} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} = \frac{3}{2} \tag{14.47}$$

Nash flow (with super road)

For a Nash flow, we equate path costs for all used paths. Checking whether all three paths are used:

Path costs at flow (α, β) with all paths used:

$$c_{P_1} = c_{s,a}(\alpha) + c_{a,t}(\alpha - \beta) = \alpha + 1 \tag{14.48}$$

$$c_{P_2} = c_{s,b}(1 - \alpha) + c_{b,t}(1 - \alpha + \beta) = 1 + (1 - \alpha + \beta) = 2 - \alpha + \beta \tag{14.49}$$

$$c_{P_3} = c_{s,a}(\alpha) + c_{a,b}(\beta) + c_{b,t}(1 - \alpha + \beta) = \alpha + 0 + (1 - \alpha + \beta) = 1 + \beta \tag{14.50}$$

Setting $c_{P_1} = c_{P_2}$:

$$\alpha + 1 = 2 - \alpha + \beta \implies 2\alpha - \beta = 1 \tag{14.51}$$

Setting $c_{P_1} = c_{P_3}$:

$$\alpha + 1 = 1 + \beta \implies \beta = \alpha \tag{14.52}$$

From $2\alpha - \alpha = 1$ we get $\alpha = 1$ and $\beta = 1$.

Checking feasibility: $\alpha - \beta = 0 \geq 0$ (path P_1 has zero flow), $1 - \alpha + \beta = 1 \geq 0$. However, $\alpha = 1$ means all flow uses $s \to a$, so we need $1 - \alpha = 0$, meaning P_2 also has zero flow. Let us verify: with $\beta = \alpha = 1$, path $P_3 = (s, a, b, t)$ carries all flow of 1.

Check: $c_{P_3} = 1 + \beta = 1 + 1 = 2$, but $c_{P_1} = \alpha + 1 = 2$ and $c_{P_2} = 2 - \alpha + \beta = 2 - 1 + 1 = 2$. All path costs equal 2.

The Nash flow is $\tilde{f} = (\alpha, \beta) = (1, 1)$, i.e. all traffic travels $s \to a \to b \to t$, with:

$$C\left(\tilde{f}\right) = 1^2 + 0 + (0) \cdot 1 + (1)^2 = 1 + 0 + 0 + 1 = 2 \tag{14.53}$$

This is Braess's Paradox: adding the zero-cost road $a \to b$ raises the Nash flow cost from $3/2$ to 2. The **Price of Anarchy** for the extended network is:

$$\text{PoA} = \frac{C\left(\tilde{f}\right)}{C(f^*)} = \frac{2}{3/2} = \frac{4}{3} \tag{14.54}$$

```python
import numpy as np
import scipy.optimize

def cost_with_super_road(f):
    alpha, beta = f
    return (alpha ** 2 + (alpha - beta) + (1 - alpha) + (1 - alpha + beta) **
2)

def potential_with_super_road(f):
    alpha, beta = f
    # int_0^alpha x dx + int_0^(alpha-beta) 1 dx + int_0^(1-alpha) 1 dx +
int_0^(1-alpha+beta) x dx + 0
    return alpha ** 2 / 2 + (alpha - beta) + (1 - alpha) + (1 - alpha + beta)
** 2 / 2
```

```
constraints = [
    {'type': 'ineq', 'fun': lambda f: f[0]},
    {'type': 'ineq', 'fun': lambda f: 1 - f[0]},
    {'type': 'ineq', 'fun': lambda f: f[1]},
    {'type': 'ineq', 'fun': lambda f: f[0] - f[1]},
    {'type': 'ineq', 'fun': lambda f: 1 - f[0] + f[1]},
]

res_opt = scipy.optimize.minimize(cost_with_super_road, [0.4, 0.1],
constraints=constraints)
res_nash = scipy.optimize.minimize(potential_with_super_road, [0.4, 0.1],
constraints=constraints)

print(f"Optimal flow (alpha, beta) = ({res_opt.x[0]:.4f}, {res_opt.x[1]:.4f}),
cost = {res_opt.fun:.4f}")
print(f"Nash flow (alpha, beta) = ({res_nash.x[0]:.4f}, {res_nash.x[1]:.4f}),
cost = {cost_with_super_road(res_nash.x):.4f}")
print(f"Price of Anarchy = {cost_with_super_road(res_nash.x) /
res_opt.fun:.4f}")
```

```
Optimal flow (alpha, beta) = (0.5000, -0.0000), cost = 1.5000
Nash flow (alpha, beta) = (1.0000, 1.0000), cost = 2.0000
Price of Anarchy = 1.3333
```

Solution 14.2: Solution to Exercise 2

We wish to verify that $\tilde{f} = (\alpha, \beta) = (1/2, 1/5)$ is a Nash flow for the delivery companies game using the potential function approach.

By Theorem: Nash Flow minimises the potential function, $\tilde{f}$ is a Nash flow if and only if it minimises the potential function:

$$\Phi(\alpha, \beta) = \frac{\alpha^3}{3} + \frac{3\beta^2}{4} + \frac{(1-\alpha-\beta)^2}{2} \tag{14.55}$$

subject to $0 \leq \alpha \leq 1/2$ and $0 \leq \beta \leq 1/2$.

Since the minimum may lie in the interior or on the boundary, we apply the KKT conditions. The feasible region is defined by:

$$g_1(\alpha, \beta) = \alpha \geq 0, \quad g_2(\alpha, \beta) = 1/2 - \alpha \geq 0, \quad g_3(\alpha, \beta) = \beta \geq 0, \quad g_4(\alpha, \beta) = 1/2 - \beta \geq 0 \tag{14.56}$$

At $(\alpha, \beta) = (1/2, 1/5)$: constraint g_2 is active ($\alpha = 1/2$), while g_1, g_3, g_4 are inactive. By complementary slackness, $\lambda_1 = \lambda_3 = \lambda_4 = 0$, and $\lambda_2 \geq 0$.

The KKT stationarity conditions are:

$$\frac{\partial \Phi}{\partial \alpha} - \lambda_2 \cdot (-1) = 0 \implies \alpha^2 - (1-\alpha-\beta) + \lambda_2 = 0 \tag{14.57}$$

$$\frac{\partial \Phi}{\partial \beta} - 0 = 0 \implies \frac{3\beta}{2} - (1-\alpha-\beta) = 0 \tag{14.58}$$

Evaluating the second condition at $(1/2, 1/5)$:

$$\frac{3}{2} \cdot \frac{1}{5} - \left(1 - \frac{1}{2} - \frac{1}{5}\right) = \frac{3}{10} - \frac{3}{10} = 0 \checkmark \tag{14.59}$$

For the first condition at $(1/2, 1/5)$:

$$\left(\frac{1}{2}\right)^2 - \left(1 - \frac{1}{2} - \frac{1}{5}\right) + \lambda_2 = \frac{1}{4} - \frac{3}{10} + \lambda_2 = -\frac{1}{20} + \lambda_2 = 0 \tag{14.60}$$

$$\implies \lambda_2 = \frac{1}{20} > 0 \checkmark \tag{14.61}$$

All KKT conditions are satisfied with $\lambda_2 = 1/20 > 0$, confirming that $(1/2, 1/5)$ is a minimum of Φ and therefore a Nash flow.

```python
import numpy as np
import scipy.optimize

def potential(f):
    alpha, beta = f
    return alpha ** 3 / 3 + 3 * beta ** 2 / 4 + (1 - alpha - beta) ** 2 / 2

constraints = [
    {'type': 'ineq', 'fun': lambda f: f[0]},
    {'type': 'ineq', 'fun': lambda f: 0.5 - f[0]},
    {'type': 'ineq', 'fun': lambda f: f[1]},
    {'type': 'ineq', 'fun': lambda f: 0.5 - f[1]},
]

res = scipy.optimize.minimize(potential, [0.3, 0.1], constraints=constraints)
print(f"Minimiser of potential: alpha = {res.x[0]:.6f}, beta =
{res.x[1]:.6f}")
print(f"Expected: alpha = 0.5, beta = 0.2")
print(f"Potential at minimiser: {res.fun:.6f}")
```

```
Minimiser of potential: alpha = 0.500000, beta = 0.200013
Expected: alpha = 0.5, beta = 0.2
Potential at minimiser: 0.116667
```

Solution 14.3: Solution to Exercise 3

We wish to confirm that

$$f = \left(\frac{\sqrt{11}}{5} - \frac{1}{5}, \frac{12}{25} - \frac{2\sqrt{11}}{25}\right) \tag{14.62}$$

is a Nash flow for the delivery companies game using the marginal cost formulation.

By Theorem: Optimal Flow is a Nash Flow for Marginal Costs, a flow is optimal for (G, r, c) if and only if it is a Nash flow for (G, r, c^*), where c^* denotes the marginal cost functions.

The original cost functions and their marginal costs are:

$$c_{s_1,t}(x) = x^2 \implies c^*_{s_1,t}(x) = \frac{d}{dx}(x \cdot x^2) = 3x^2 \tag{14.63}$$

$$c_{s_2,t}(x) = \frac{3}{2}x \implies c^*_{s_2,t}(x) = \frac{d}{dx}\left(\frac{3}{2}x^2\right) = 3x \tag{14.64}$$

$$c_{a,t}(x) = x \implies c^*_{a,t}(x) = \frac{d}{dx}(x^2) = 2x \tag{14.65}$$

$$c_{s_1,a}(x) = 0 \implies c^*_{s_1,a}(x) = 0, \qquad c_{s_2,a}(x) = 0 \implies c^*_{s_2,a}(x) = 0 \tag{14.66}$$

With $\alpha = \frac{\sqrt{11}}{5} - \frac{1}{5}$ and $\beta = \frac{12}{25} - \frac{2\sqrt{11}}{25}$, the shared edge $a \to t$ carries flow $f_{a,t} = 1 - \alpha - \beta$. The quadratic $5\alpha^2 + 2\alpha - 2 = 0$ has roots $\left(-1 \pm \sqrt{11}\right)/5$, and we take the positive root. Numerically,

$$\alpha \approx 0.4633, \qquad \beta \approx 0.2147, \qquad 1 - \alpha - \beta \approx 0.3220, \tag{14.67}$$

so all three flows are positive and the flow is feasible.

To verify f^* is a Nash flow for the marginal cost game, we need to check that all used paths have equal marginal cost. The paths for commodity 1 are $P_{s_1,t} = (s_1, t)$ and $P_{s_1,a,t} = (s_1, a, t)$.

The marginal cost along $P_{s_1,t}$ is $c^*_{s_1,t}(\alpha) = 3\alpha^2$.

The marginal cost along $P_{s_1,a,t}$ is $c^*_{s_1,a}(0) + c^*_{a,t}(1 - \alpha - \beta) = 0 + 2(1 - \alpha - \beta)$.

At the optimal flow, these must be equal (since the optimal flow uses both paths for commodity 1):

$$3\alpha^2 = 2(1 - \alpha - \beta) \tag{14.68}$$

This is exactly the first stationarity condition from the optimal flow derivation. Similarly, for commodity 2 using both its paths $P_{s_2,t} = (s_2, t)$ and $P_{s_2,a,t} = (s_2, a, t)$:

Marginal cost along $P_{s_2,t}$: $c^*_{s_2,t}(\beta) = 3\beta$.

Marginal cost along $P_{s_2,a,t}$: $c^*_{s_2,a}(0) + c^*_{a,t}(1 - \alpha - \beta) = 2(1 - \alpha - \beta)$.

At the optimal flow, $3\beta = 2(1 - \alpha - \beta)$, which is the second stationarity condition.

Since f^* was found precisely by solving these two equations simultaneously, it satisfies the Nash flow conditions for (G, r, c^*). By Theorem: Optimal Flow is a Nash Flow for Marginal Costs, f^* is therefore a Nash flow for the marginal cost game.

```python
import sympy as sym

alpha, beta = sym.symbols('alpha beta', real=True)

# Optimal flow conditions: 3*alpha^2 = 2*(1 - alpha - beta) and 3*beta = 2*(1
- alpha - beta)
eq1 = sym.Eq(3 * alpha**2, 2 * (1 - alpha - beta))
eq2 = sym.Eq(3 * beta, 2 * (1 - alpha - beta))

solutions = sym.solve([eq1, eq2], [alpha, beta])
print("Solutions (alpha, beta):")
for sol in solutions:
    a_val, b_val = sol
    print(f"  alpha = {a_val} ≈ {float(a_val):.6f},  beta = {b_val} ≈
{float(b_val):.6f}")
    print(f"  Feasible (both in [0,1/2])? alpha>=0: {float(a_val) >= 0},
beta>=0: {float(b_val) >= 0}")
```

```
Solutions (alpha, beta):
  alpha = -1/5 + sqrt(11)/5 ≈ 0.463325,  beta = 12/25 - 2*sqrt(11)/25 ≈
0.214670
  Feasible (both in [0,1/2])? alpha>=0: True, beta>=0: True
  alpha = -sqrt(11)/5 - 1/5 ≈ -0.863325,  beta = 2*sqrt(11)/25 + 12/25 ≈
0.745330
  Feasible (both in [0,1/2])? alpha>=0: False, beta>=0: True
```

```python
# Verify: at the feasible solution, marginal costs on used paths are equal
a_val = solutions[1][0]  # take the positive root
b_val = solutions[1][1]

shared_flow = 1 - a_val - b_val
mc_P1 = 3 * a_val**2
mc_P1_alt = 2 * shared_flow
mc_P2 = 3 * b_val
mc_P2_alt = 2 * shared_flow

print(f"Marginal cost on (s1,t): {sym.simplify(mc_P1)}")
print(f"Marginal cost on (s1,a,t): {sym.simplify(mc_P1_alt)}")
print(f"Equal? {sym.simplify(mc_P1 - mc_P1_alt) == 0}")
print(f"\nMarginal cost on (s2,t): {sym.simplify(mc_P2)}")
print(f"Marginal cost on (s2,a,t): {sym.simplify(mc_P2_alt)}")
print(f"Equal? {sym.simplify(mc_P2 - mc_P2_alt) == 0}")
```

```
Marginal cost on (s1,t): 6*sqrt(11)/25 + 36/25
Marginal cost on (s1,a,t): 6*sqrt(11)/25 + 36/25
Equal? True

Marginal cost on (s2,t): 6*sqrt(11)/25 + 36/25
Marginal cost on (s2,a,t): 6*sqrt(11)/25 + 36/25
Equal? True
```

15. Matching Games

Many allocation problems require pairing agents from two groups such that no pair would mutually prefer to be matched to each other instead. This chapter develops the theory of stable matchings and the Gale–Shapley algorithm that constructs them.

15.1 Motivating Example: Stable peer review assignment at a research conference

The **Social Perspectives Research Symposium** receives five paper submissions and recruits five expert reviewers. Each paper must be assigned to exactly one reviewer.

To promote high-quality reviews, both authors and reviewers submit ranked preference lists:

- Each **author** ranks the reviewers based on perceived expertise, familiarity with the topic, and potential for constructive feedback.
- Each **reviewer** ranks the submissions based on interest, alignment with their research, and methodological fit.

The participants are:

- **Authors:** A1, A2, A3, A4, A5
- **Reviewers:** R1, R2, R3, R4, R5

For example:

- Author A1 has written a paper on representation in environmental activism and ranks R3, an expert in environmental sociology, as their top choice.
- Reviewer R2 specializes in healthcare policy and ranks submission A4 as their first choice due to its relevance.
- Author A5's paper focuses on queer representation in contemporary fiction, and they rank R5 highly based on prior work in literature studies.

"R3's expertise would be invaluable for my analysis."
"A4's topic fits directly with my current research."
"R5's previous reviews have been extremely helpful."

The preferences of the authors are given in Table 15.1 and the preferences of the reviewers are given in Table 15.2.

Author	1st	2nd	3rd	4th	5th
A1	R3	R1	R4	R5	R2
A2	R2	R4	R3	R5	R1
A3	R5	R2	R3	R1	R4
A4	R2	R3	R5	R1	R4
A5	R5	R3	R4	R2	R1

Table 15.1: Preference table for the authors

The organizers aim to compute a **stable matching**: an assignment where no author-reviewer pair exists who would both prefer to be assigned to each other over their current assignments.

Reviewer	1st	2nd	3rd	4th	5th
R1	A3	A1	A5	A4	A2

R2	A4	A2	A5	A1	A3
R3	A1	A5	A2	A3	A4
R4	A2	A5	A3	A1	A4
R5	A5	A3	A4	A1	A2

Table 15.2: Preference table for the reviewers

15.2 Theory

15.2.1 Definition: Matching Game

A matching game of size N is defined by two disjoint sets S and R of proposers and reviewers of size N. Associated to each element of S and R is a preference map:

$$f : S \to R^N \quad \text{and } g : R \to S^N \tag{15.1}$$

A matching M is any bijection between S and R. If $s \in S$ and $r \in R$ are matched by M we denote:

$$M(s) = r \tag{15.2}$$

15.2.1.1 Example: Author Reviewer as a Matching Game

For Motivating Example: Stable peer review assignment at a research conference the matching game has size $N = 5$, the set of proposers $S = (A1, A2, A3, A4, A5)$ the set of reviewers $R = (R1, R2, R3, R4, R5)$ and the preference maps:

$$\begin{aligned}
f(A1) &= (R3, R1, R4, R5, R2) \\
f(A2) &= (R2, R4, R3, R5, R1) \\
f(A3) &= (R5, R2, R3, R1, R4) \\
f(A4) &= (R2, R3, R5, R1, R4) \\
f(A5) &= (R5, R3, R4, R2, R1) \\
g(R1) &= (A3, A1, A5, A4, A2) \\
g(R2) &= (A4, A2, A5, A1, A3) \\
g(R3) &= (A1, A5, A2, A3, A4) \\
g(R4) &= (A2, A5, A3, A1, A4) \\
g(R5) &= (A5, A3, A4, A1, A2)
\end{aligned} \tag{15.3}$$

One potential mapping is given by:

$$\begin{aligned}
M(A1) &= R3 \\
M(A2) &= R2 \\
M(A3) &= R5 \\
M(A4) &= R1 \\
M(A5) &= R4
\end{aligned} \tag{15.4}$$

15.2.2 Definition: a blocking pair

A pair (s, r) is said to **block** a matching M if $M(s) \neq r$ but s prefers r to $M(s)$ and r prefers s to $M^{-1}(r)$.

15.2.2.1 Example: Blocking pair for the author reviewer game

In (15.4) the pair $(A4, R2)$ is a blocking pair as $A4$ prefer $R2$ to $M(A4)$ and $R2$ prefers $A4$ to $M^{-1}(R2)$.

15.2.3 Definition: a stable matching

A matching M with no blocking pair is said to be stable.

15.2.3.1 Example: A stable matching for the author reviewer game

The following is a stable matching for Motivating Example: Stable peer review assignment at a research conference:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R4 \\ M(A3) &= R1 \\ M(A4) &= R2 \\ M(A5) &= R5 \end{aligned} \tag{15.5}$$

15.2.4 Definition: The Gale-Shapley algorithm

Here is the Gale-Shapley algorithm, which gives a stable matching for a matching game:

1. Assign every $s \in S$ and $r \in R$ to be unmatched
2. Pick some unmatched $s \in S$, let r be the top of s's preference list:
    - If r is unmatched set $M(s) = r$
    - If r is matched:
        - If r prefers s to $M^{-1}(r)$ then set $M(s) = r$ and unmatch the previous partner $M^{-1}(r)$
        - Otherwise s remains unmatched and remove r from s's preference list.
3. Repeat step 2 until all $s \in S$ are matched.

15.2.5 Theorem: Unique matching as output of the Gale Shapley algorithm

All possible executions of the Gale-Shapley algorithm yield the same stable matching **and** in this stable matching every suitor has the best possible partner in any stable matching.

15.2.5.1 Proof

Call a reviewer r a **stable partner** of a suitor s if $M''(s) = r$ in some stable matching M''. We show that no suitor is ever rejected by a stable partner during an execution α. Since suitors propose in decreasing order of preference, this implies that each suitor ends α matched to their most preferred stable partner, so the outcome does not depend on α.

Suppose, for contradiction, that during α some suitor is rejected by a stable partner, and consider the **first** such rejection: reviewer r' rejects suitor s, where r' is a stable partner of s, so $M'(s) = r'$ for some stable matching M'. The reviewer r' rejects s in favour of a suitor s' whom r' prefers to s; that is, at that moment r' holds a proposal from s' and prefers s' to s.

Because this is the first rejection by a stable partner, s' has not yet been rejected by any stable partner. As suitors propose in order of preference and s' is currently proposing to r', the reviewer r' is at least as high in s''s list as any reviewer s' has proposed to so far, and in particular at least as high as every stable partner of s'. Hence s' weakly prefers r' to $M'(s')$, and since preferences are strict and $M'(s') \neq r'$ (as $M'(r') = s$), s' strictly prefers r' to $M'(s')$.

Then s' prefers r' to $M'(s')$ and r' prefers s' to $s = M'(r')$, so (s', r') blocks M', contradicting the stability of M'. No suitor is therefore rejected by a stable partner, each suitor is matched in M with their favourite stable reviewer, and since α was arbitrary all executions give the same matching.

15.2.5.2 Example: Application of the Gale-Shapley algorithm to the author reviewer game

We start by assigning $A1, A2, A3, A4, A5$ and $R1, R2, R3, R4, R5$ to be unmatched.

We pick $A1$ which has $f(A1) = (R3, R1, R4, R5, R2)$, the top of the preference list is $R3$ so we set:

$$M(A1) = R3 \tag{15.6}$$

We now pick $A2$ which has $f(A2) = (R2, R4, R3, R5, R1)$, the top of the preference list is $R2$ so we set:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R2 \end{aligned} \tag{15.7}$$

We now pick $A3$ which has $f(A3) = (R5, R2, R3, R1, R4)$, the top of the preference list is $R5$ so we set:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R2 \\ M(A3) &= R5 \end{aligned} \tag{15.8}$$

We now pick $A4$ which has $f(A4) = (R2, R3, R5, R1, R4)$, the top of the preference list is $R2$ but $R2$ is matched ($M(A2) = R2$). We have $g(R2) = (A4, A2, A5, A1, A3)$ so $R2$ prefers $A4$ to their current matching $A2$. So $A2$ becomes unmatched and we set:

$$\begin{aligned} M(A1) &= R3 \\ M(A3) &= R5 \\ M(A4) &= R2 \end{aligned} \tag{15.9}$$

We now pick $A2$ (for the second time) which has $f(A2) = (R2, R4, R3, R5, R1)$, the top of the preference list is $R2$ but $R2$ is matched and prefers their current matching so we remove it from $A2$'s preference list:

$$f(A2) = (R4, R3, R5, R1) \tag{15.10}$$

We could pick any unmatched author, we will pick $A2$ again (for the third time), $A2$ has preference list $f(A2) = (R4, R3, R5, R1)$, the top of the preference list is $R4$ so we set:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R4 \\ M(A3) &= R5 \\ M(A4) &= R2 \end{aligned} \tag{15.11}$$

We now pick $A5$, the final unmatched author, which has $f(A5) = (R5, R3, R4, R2, R1)$, the top of the preference list is $R5$ but $R5$ is matched ($M(A3) = R5$). We have $g(R5) = (A5, A3, A4, A1, A2)$ so $R5$ prefers $A5$ to their current matching. So we set:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R4 \\ M(A4) &= R2 \\ M(A5) &= R5 \end{aligned} \tag{15.12}$$

We now pick $A3$, the one remaining unmatched author, which has $f(A3) = (R5, R2, R3, R1, R4)$, the top of the preference list is $R5$ but $R5$ is matched and prefers their current matching so we remove it from $A3$'s preference list:

$$f(A3) = (R2, R3, R1, R4) \tag{15.13}$$

We now pick $A3$ again as it is still the only unmatched author. The top of the preference list is $R2$ but $R2$ is matched and prefers their current matching so we remove it from $A3$'s preference list:

$$f(A3) = (R3, R1, R4) \tag{15.14}$$

We pick $A3$ again. The top of the preference list is $R3$ but $R3$ is matched and prefers their current matching so we remove it from $A3$'s preference list:

$$f(A3) = (R1, R4) \tag{15.15}$$

We pick $A3$ again and match it to $R1$ giving the final matching:

$$\begin{aligned} M(A1) &= R3 \\ M(A2) &= R4 \\ M(A3) &= R1 \\ M(A4) &= R2 \\ M(A5) &= R5 \end{aligned} \tag{15.16}$$

15.2.6 Theorem: Reviewer sub optimality of the Gale Shapley Algorithm

In a suitor-optimal stable matching each reviewer has the worst possible matching.

15.2.6.1 Proof

Assume that the result is not true. Let M_0 be a suitor-optimal matching and assume that there is a stable matching M' such that $\exists\, r$ such that r prefers $s = M_0^{-1}(r)$ to $s' = M'^{-1}(r)$. Since preferences are strict and $M'(s) \neq r$, the suitor s either prefers $M'(s)$ to $M_0(s) = r$ or prefers $M_0(s) = r$ to $M'(s)$. The former is impossible, as suitor-optimality means s has no stable match they prefer to $M_0(s)$. Hence s prefers r to $M'(s)$, and as r prefers s to $M'(r) = s'$, the pair (r, s) blocks M', contradicting its stability. Every reviewer therefore does at least as badly in M_0 as in any stable matching, so M_0 is reviewer-pessimal.

15.3 Exercises

Exercise 15.1:

Obtain stable suitor optimal and reviewer optimal matchings for the games shown:

1.

$$\begin{aligned} f(A1) &= (R2, R1, R3) \\ f(A2) &= (R1, R3, R2) \\ f(A3) &= (R1, R3, R2) \\ g(R1) &= (A1, A2, A3) \\ g(R2) &= (A2, A1, A3) \\ g(R3) &= (A2, A3, A1) \end{aligned} \tag{15.17}$$

1.

$$
\begin{aligned}
f(A1) &= (R1, R3, R2) \\
f(A2) &= (R2, R3, R1) \\
f(A3) &= (R1, R3, R2) \\
g(R1) &= (A1, A2, A3) \\
g(R2) &= (A2, A3, A1) \\
g(R3) &= (A2, A3, A1)
\end{aligned}
\tag{15.18}
$$

1.

$$
\begin{aligned}
f(A1) &= (R1, R4, R2, R3) \\
f(A2) &= (R2, R1, R3, R4) \\
f(A3) &= (R4, R1, R3, R2) \\
f(A4) &= (R1, R4, R3, R2) \\
g(R1) &= (A1, A4, A2, A3) \\
g(R2) &= (A1, A3, A4, A2) \\
g(R3) &= (A4, A1, A3, A2) \\
g(R4) &= (A2, A4, A1, A3)
\end{aligned}
\tag{15.19}
$$

1.

$$
\begin{aligned}
f(A1) &= (R2, R1, R4, R3) \\
f(A2) &= (R2, R3, R1, R4) \\
f(A3) &= (R1, R4, R3, R2) \\
f(A4) &= (R1, R4, R3, R2) \\
g(R1) &= (A1, A2, A4, A3) \\
g(R2) &= (A4, A2, A1, A3) \\
g(R3) &= (A4, A1, A3, A2) \\
g(R4) &= (A3, A2, A4, A1)
\end{aligned}
\tag{15.20}
$$

Exercise 15.2:

Consider a matching game where all reviewers have the same preference list. Prove that there is a single stable matching.

15.4 Programming

The python Matching library (Wilde et al., 2020) implements the Gale-Shapley algorithm as well as a number of other algorithms for different generalisations of matching games.

```
import matching
import matching.games

authors = [
    matching.Player("A1"),
    matching.Player("A2"),
    matching.Player("A3"),
```

```
    matching.Player("A4"),
    matching.Player("A5"),
]

reviewers = [
    matching.Player("R1"),
    matching.Player("R2"),
    matching.Player("R3"),
    matching.Player("R4"),
    matching.Player("R5"),
]

A1, A2, A3, A4, A5 = authors
R1, R2, R3, R4, R5 = reviewers

A1.set_prefs((R3, R1, R4, R5, R2))
A2.set_prefs((R2, R4, R3, R5, R1))
A3.set_prefs((R5, R2, R3, R1, R4))
A4.set_prefs((R2, R3, R5, R1, R4))
A5.set_prefs((R5, R3, R4, R2, R1))

R1.set_prefs((A3, A1, A5, A4, A2))
R2.set_prefs((A4, A2, A5, A1, A3))
R3.set_prefs((A1, A5, A2, A3, A4))
R4.set_prefs((A2, A5, A3, A1, A4))
R5.set_prefs((A5, A3, A4, A1, A2))

game = matching.games.StableMarriage(authors, reviewers)
print(f"Stable matching: {game.solve()}")
```

```
Stable matching: {A1: R3, A2: R4, A3: R1, A4: R2, A5: R5}
```

15.5 Notable Research

The original Gale-Shapley algorithm was presented in (Gale & Shapley, 1962), a rigorous combinatorial treatment of the algorithm is given in (Knuth, 1976). Some refinements to the algorithm are given in (Gusfield & Irving, 1989) which is considered to be the reference text on matching games.

In the original paper (Gale & Shapley, 1962) Gale and Shapley present the so called hospital-resident matching problem which is a matching problem aiming to match many to one. They do so in the context of College admissions in North America. In (Roth, 1984) presents the problem in the context of hospital resident assignement. Further to this, Roth considered the problem of kidney exchange in (Roth et al., 2004) as a further matching problem.

In 2012, Lloyd Shapley and Alvin Roth were awarded the Nobel Prize in Economic Sciences for their contributions to stable matching and market design. David Gale, who co-authored the foundational 1962 paper introducing the Gale-Shapley algorithm, was ineligible for the award, having passed away in 2008.

15.6 Conclusion

In this chapter we introduced the theory of matching games, motivated by the practical problem of assigning reviewers to papers at a research conference. The central concept of **stability** ensures that no pair of participants would prefer to deviate from the assignment, making stable matchings attractive in many real-world settings.

The Gale-Shapley algorithm provides an elegant and efficient solution to the stable matching problem, always producing a suitor-optimal stable matching. While stable matchings always exist in the one-to-one setting, extensions to many-to-one settings (such as hospital-resident assignments) preserve many of the desirable properties of the original algorithm.

A summary of key concepts introduced in this chapter is given in Table 15.3.

Concept	Description
Matching Game	A game where two disjoint sets have preferences over each other
Stable Matching	A matching with no blocking pairs
Blocking Pair	A pair who would both prefer to be matched to each other over their current assignments
Gale-Shapley Algorithm	An algorithm that produces a stable matching by iteratively proposing and accepting/rejecting proposals
Suitor-optimality	The property that each suitor receives the best possible partner across all stable matchings
Reviewer-pessimality	The property that in suitor-optimal matchings, each reviewer receives the worst acceptable partner

Table 15.3: Summary of key concepts in matching games

Important

The Gale-Shapley algorithm demonstrates how a simple, local, step-by-step procedure can achieve globally stable outcomes even in complex preference structures. This insight has led to numerous successful applications in market design, from medical residency matches to school choice systems and organ donation programs.

15.7 Solutions

Solution 15.1: Solution to Exercise 1

For each game we run the Gale-Shapley algorithm from the suitor (A) side to obtain the suitor-optimal matching, then repeat with the reviewer (R) side proposing to obtain the reviewer-optimal (equivalently, suitor-pessimal) matching.

1.

$$\begin{aligned} f(A1) &= (R2, R1, R3) \\ f(A2) &= (R1, R3, R2) \\ f(A3) &= (R1, R3, R2) \\ g(R1) &= (A1, A2, A3) \\ g(R2) &= (A2, A1, A3) \\ g(R3) &= (A2, A3, A1) \end{aligned} \tag{15.21}$$

Suitor-optimal matching (A proposes):

- A1 proposes to R2 (top of A1's list). R2 is unmatched, so $M(A1) = R2$.
- A2 proposes to R1 (top of A2's list). R1 is unmatched, so $M(A2) = R1$.

- A3 proposes to R1 (top of A3′s list). R1 is matched to A2. $g(R1) = (A1, A2, A3)$, so R1 prefers A2 to A3. A3 is rejected; remove R1 from A3′s list: $f(A3) = (R3, R2)$.
- A3 proposes to R3. R3 is unmatched, so $M(A3) = R3$.

Suitor-optimal matching:

$$M_S: \quad M(A1) = R2, \quad M(A2) = R1, \quad M(A3) = R3 \tag{15.22}$$

Verification (no blocking pairs):

- $(A1, R1)$: A1 prefers R2 to R1 (R2 is first in A1′s list). Not blocking.
- $(A1, R3)$: A1 prefers R2 to R3. Not blocking.
- $(A2, R2)$: A2 prefers R1 to R2. Not blocking.
- $(A2, R3)$: A2 prefers R1 to R3. Not blocking.
- $(A3, R1)$: A3 prefers R1 to R3, but R1 prefers A2 (current match) to A3. Not blocking.
- $(A3, R2)$: A3 prefers R3 to R2 (R3 is second, R2 is third in A3′s remaining list). Actually $f(A3) = (R1, R3, R2)$ originally, so A3 prefers R3 to R2. Not blocking.

The matching is stable.

Reviewer-optimal matching (R proposes):

- R1 proposes to A1 (top of R1′s list). A1 is unmatched, so $M^{-1}(R1) = A1$.
- R2 proposes to A2 (top of R2′s list). A2 is unmatched, so $M^{-1}(R2) = A2$.
- R3 proposes to A2 (top of R3′s list). A2 is matched to R2. $f(A2) = (R1, R3, R2)$, so A2 prefers R3 to R2. R2 is rejected; $M^{-1}(R3) = A2$. R2 removes A2 from its list: $g(R2) = (A1, A3)$.
- R2 proposes to A1. A1 is matched to R1. $f(A1) = (R2, R1, R3)$, so A1 prefers R2 to R1. R1 is rejected; $M^{-1}(R2) = A1$. R1 removes A1 from its list: $g(R1) = (A2, A3)$.
- R1 proposes to A2. A2 is matched to R3. $f(A2) = (R1, R3, R2)$, so A2 prefers R1 to R3. R3 is rejected; $M^{-1}(R1) = A2$. R3 removes A2 from its list: $g(R3) = (A3, A1)$.
- R3 proposes to A3. A3 is unmatched, so $M^{-1}(R3) = A3$.

Reviewer-optimal matching:

$$M_R: \quad M(A1) = R2, \quad M(A2) = R1, \quad M(A3) = R3 \tag{15.23}$$

In this case both matchings coincide.

2.

$$\begin{aligned} f(A1) &= (R1, R3, R2) \\ f(A2) &= (R2, R3, R1) \\ f(A3) &= (R1, R3, R2) \\ g(R1) &= (A1, A2, A3) \\ g(R2) &= (A2, A3, A1) \\ g(R3) &= (A2, A3, A1) \end{aligned} \tag{15.24}$$

Suitor-optimal matching (A proposes):

- A1 proposes to R1. R1 is unmatched, so $M(A1) = R1$.
- A2 proposes to R2. R2 is unmatched, so $M(A2) = R2$.

- A3 proposes to R1. R1 is matched to A1. $g(R1) = (A1, A2, A3)$, so R1 prefers A1 to A3. A3 is rejected; $f(A3) = (R3, R2)$.
- A3 proposes to R3. R3 is unmatched, so $M(A3) = R3$.

Suitor-optimal matching:

$$M_S: \quad M(A1) = R1, \quad M(A2) = R2, \quad M(A3) = R3 \tag{15.25}$$

Reviewer-optimal matching (R proposes):

- R1 proposes to A1. A1 is unmatched, so $M^{-1}(R1) = A1$.
- R2 proposes to A2. A2 is unmatched, so $M^{-1}(R2) = A2$.
- R3 proposes to A2. A2 is matched to R2. $f(A2) = (R2, R3, R1)$, so A2 prefers R2 to R3. R3 is rejected; $g(R3) = (A3, A1)$.
- R3 proposes to A3. A3 is unmatched, so $M^{-1}(R3) = A3$.

Reviewer-optimal matching:

$$M_R: \quad M(A1) = R1, \quad M(A2) = R2, \quad M(A3) = R3 \tag{15.26}$$

Again both matchings coincide.

3.

$$\begin{aligned} f(A1) &= (R1, R4, R2, R3) \\ f(A2) &= (R2, R1, R3, R4) \\ f(A3) &= (R4, R1, R3, R2) \\ f(A4) &= (R1, R4, R3, R2) \\ g(R1) &= (A1, A4, A2, A3) \\ g(R2) &= (A1, A3, A4, A2) \\ g(R3) &= (A4, A1, A3, A2) \\ g(R4) &= (A2, A4, A1, A3) \end{aligned} \tag{15.27}$$

Suitor-optimal matching (A proposes):

- A1 proposes to R1. R1 is unmatched, so $M(A1) = R1$.
- A2 proposes to R2. R2 is unmatched, so $M(A2) = R2$.
- A3 proposes to R4. R4 is unmatched, so $M(A3) = R4$.
- A4 proposes to R1. R1 matched to A1. $g(R1) = (A1, A4, A2, A3)$: R1 prefers A1 to A4. A4 rejected; $f(A4) = (R4, R3, R2)$.
- A4 proposes to R4. R4 matched to A3. $g(R4) = (A2, A4, A1, A3)$: R4 prefers A4 to A3. A3 displaced; $M(A4) = R4$. A3′s list: $f(A3) = (R1, R3, R2)$.
- A3 proposes to R1. R1 matched to A1. $g(R1) = (A1, A4, A2, A3)$: R1 prefers A1 to A3. A3 rejected; $f(A3) = (R3, R2)$.
- A3 proposes to R3. R3 unmatched, so $M(A3) = R3$.

Suitor-optimal matching:

$$M_S: \quad M(A1) = R1, \quad M(A2) = R2, \quad M(A3) = R3, \quad M(A4) = R4 \tag{15.28}$$

Reviewer-optimal matching (R proposes):

- R1 proposes to A1. A1 unmatched, so $M^{-1}(R1) = A1$.

- R2 proposes to A1. A1 matched to R1. $f(A1) = (R1, R4, R2, R3)$: A1 prefers R1 to R2. R2 rejected; $g(R2) = (A3, A4, A2)$.
- R3 proposes to A4. A4 unmatched, so $M^{-1}(R3) = A4$.
- R4 proposes to A2. A2 unmatched, so $M^{-1}(R4) = A2$.
- R2 proposes to A3. A3 unmatched, so $M^{-1}(R2) = A3$.

Reviewer-optimal matching:

$$M_R: \quad M(A1) = R1, \quad M(A2) = R4, \quad M(A3) = R2, \quad M(A4) = R3 \tag{15.29}$$

4.

$$\begin{aligned} f(A1) &= (R2, R1, R4, R3) \\ f(A2) &= (R2, R3, R1, R4) \\ f(A3) &= (R1, R4, R3, R2) \\ f(A4) &= (R1, R4, R3, R2) \\ g(R1) &= (A1, A2, A4, A3) \\ g(R2) &= (A4, A2, A1, A3) \\ g(R3) &= (A4, A1, A3, A2) \\ g(R4) &= (A3, A2, A4, A1) \end{aligned} \tag{15.30}$$

Suitor-optimal matching (A proposes):

- A1 proposes to R2. R2 unmatched, so $M(A1) = R2$.
- A2 proposes to R2. R2 matched to A1. $g(R2) = (A4, A2, A1, A3)$: R2 prefers A2 to A1. A1 displaced; $M(A2) = R2$. A1′s list: $f(A1) = (R1, R4, R3)$.
- A3 proposes to R1. R1 unmatched, so $M(A3) = R1$.
- A4 proposes to R1. R1 matched to A3. $g(R1) = (A1, A2, A4, A3)$: R1 prefers A4 to A3. A3 displaced; $M(A4) = R1$. A3′s list: $f(A3) = (R4, R3, R2)$.
- A1 proposes to R1. R1 matched to A4. $g(R1) = (A1, A2, A4, A3)$: R1 prefers A1 to A4. A4 displaced; $M(A1) = R1$. A4′s list: $f(A4) = (R4, R3, R2)$.
- A4 proposes to R4. R4 unmatched, so $M(A4) = R4$.
- A3 proposes to R4. R4 matched to A4. $g(R4) = (A3, A2, A4, A1)$: R4 prefers A3 to A4. A4 displaced; $M(A3) = R4$. A4′s list: $f(A4) = (R3, R2)$.
- A4 proposes to R3. R3 unmatched, so $M(A4) = R3$.

Suitor-optimal matching:

$$M_S: \quad M(A1) = R1, \quad M(A2) = R2, \quad M(A3) = R4, \quad M(A4) = R3 \tag{15.31}$$

Reviewer-optimal matching (R proposes):

- R1 proposes to A1. A1 unmatched, so $M^{-1}(R1) = A1$.
- R2 proposes to A4. A4 unmatched, so $M^{-1}(R2) = A4$.
- R3 proposes to A4. A4 matched to R2. $f(A4) = (R1, R4, R3, R2)$: A4 prefers R3 to R2. R2 rejected; $M^{-1}(R3) = A4$. $g(R2) = (A2, A1, A3)$.
- R4 proposes to A3. A3 unmatched, so $M^{-1}(R4) = A3$.
- R2 proposes to A2. A2 unmatched, so $M^{-1}(R2) = A2$.

Reviewer-optimal matching:

$$M_R: \quad M(A1) = R1, \quad M(A2) = R2, \quad M(A3) = R4, \quad M(A4) = R3 \tag{15.32}$$

Both matchings again coincide.

The following code verifies the suitor-optimal matchings using the `matching` library:

```
import matching
import matching.games


def solve_matching(suitor_prefs, reviewer_prefs):
    suitor_names = list(suitor_prefs.keys())
    reviewer_names = list(reviewer_prefs.keys())
    suitors = {name: matching.Player(name) for name in suitor_names}
    reviewers = {name: matching.Player(name) for name in reviewer_names}
    for name, prefs in suitor_prefs.items():
        suitors[name].set_prefs([reviewers[r] for r in prefs])
    for name, prefs in reviewer_prefs.items():
        reviewers[name].set_prefs([suitors[s] for s in prefs])
    game = matching.games.StableMarriage(
        list(suitors.values()), list(reviewers.values())
    )
    result = game.solve()
    return {str(k): str(v) for k, v in result.items()}


# Game 1
result1 = solve_matching(
    {"A1": ["R2","R1","R3"], "A2": ["R1","R3","R2"], "A3": ["R1","R3","R2"]},
    {"R1": ["A1","A2","A3"], "R2": ["A2","A1","A3"], "R3": ["A2","A3","A1"]},
)
print("Game 1 suitor-optimal:", result1)
```

```
Game 1 suitor-optimal: {'A1': 'R2', 'A2': 'R1', 'A3': 'R3'}
```

```
# Game 2
result2 = solve_matching(
    {"A1": ["R1","R3","R2"], "A2": ["R2","R3","R1"], "A3": ["R1","R3","R2"]},
    {"R1": ["A1","A2","A3"], "R2": ["A2","A3","A1"], "R3": ["A2","A3","A1"]},
)
print("Game 2 suitor-optimal:", result2)
```

```
Game 2 suitor-optimal: {'A1': 'R1', 'A2': 'R2', 'A3': 'R3'}
```

```
# Game 3
result3 = solve_matching(
    {
        "A1": ["R1","R4","R2","R3"],
        "A2": ["R2","R1","R3","R4"],
        "A3": ["R4","R1","R3","R2"],
        "A4": ["R1","R4","R3","R2"],
    },
    {
        "R1": ["A1","A4","A2","A3"],
```

```
        "R2": ["A1","A3","A4","A2"],
        "R3": ["A4","A1","A3","A2"],
        "R4": ["A2","A4","A1","A3"],
    },
)
print("Game 3 suitor-optimal:", result3)
```

```
Game 3 suitor-optimal: {'A1': 'R1', 'A2': 'R2', 'A3': 'R3', 'A4': 'R4'}
```

```
# Game 4
result4 = solve_matching(
    {
        "A1": ["R2","R1","R4","R3"],
        "A2": ["R2","R3","R1","R4"],
        "A3": ["R1","R4","R3","R2"],
        "A4": ["R1","R4","R3","R2"],
    },
    {
        "R1": ["A1","A2","A4","A3"],
        "R2": ["A4","A2","A1","A3"],
        "R3": ["A4","A1","A3","A2"],
        "R4": ["A3","A2","A4","A1"],
    },
)
print("Game 4 suitor-optimal:", result4)
```

```
Game 4 suitor-optimal: {'A1': 'R1', 'A2': 'R2', 'A3': 'R4', 'A4': 'R3'}
```

Solution 15.2: Solution to Exercise 2

Claim: If all reviewers share the same preference list over suitors, there is exactly one stable matching.

Proof:

Let the shared reviewer preference list rank the suitors as $s_{\sigma(1)} \succ s_{\sigma(2)} \succ \cdots \succ s_{\sigma(N)}$ (where σ is a permutation of $\{1, \ldots, N\}$ giving the common ordering).

We argue by induction on the number of suitors that the stable matching is unique. The base case of a single suitor and reviewer is immediate.

Consider the top-ranked suitor $s_{\sigma(1)}$, whom every reviewer prefers to every other suitor. Let $r^\star$ be $s_{\sigma(1)}$'s own most preferred reviewer. In **any** stable matching M, the suitor $s_{\sigma(1)}$ is matched to $r^\star$: if instead $M\left(s_{\sigma(1)}\right) \neq r^\star$, then $s_{\sigma(1)}$ prefers $r^\star$ to $M\left(s_{\sigma(1)}\right)$, and $r^\star$ prefers $s_{\sigma(1)}$ to its own partner $M^{-1}(r^\star)$ (as $r^\star$, like every reviewer, ranks $s_{\sigma(1)}$ first), so $\left(s_{\sigma(1)}, r^\star\right)$ blocks M. Hence every stable matching pairs $s_{\sigma(1)}$ with $r^\star$.

Removing $s_{\sigma(1)}$ and $r^\star$ leaves an instance on $N-1$ suitors in which the reviewers still share a common ranking (the restriction of the original one). A matching of the full instance is stable if and only if it pairs $s_{\sigma(1)}$ with $r^\star$ and restricts to a stable matching of the smaller instance: no pair involving $s_{\sigma(1)}$ or $r^\star$ can block, since each has its most preferred partner, and any other blocking

pair would block the smaller instance too. By the induction hypothesis the smaller instance has a unique stable matching, so the full instance does as well.

Hence there is exactly one stable matching. ⬛

16. Auctions

Auctions allocate goods to the bidders who value them most, without the seller needing to know those values. This chapter analyses first- and second-price sealed-bid auctions, deriving equilibrium bidding strategies and the surprising revenue equivalence between the two formats.

This is the one chapter in which players hold private information, here each bidder's own valuation, so the games are games of **incomplete information**. The appropriate solution concept is then the Bayesian Nash equilibrium, in which each bidder best responds in expectation over the others' unknown valuations. The rest of the book works with complete information, where payoffs are common knowledge. A fuller treatment of Bayesian games and mechanism design is beyond our scope; we use auctions as a first, self-contained encounter with strategic reasoning under private information.

16.1 Motivating Example: Bidding for backstage passes

A popular band, **The Algorithms**, is playing a one-night-only concert in Cardiff.
To raise money for charity, the band is auctioning off a single **backstage pass**.

> Three fans (Alex, Casey, and Jordan) are each secretly asked to submit a sealed bid for how much they are willing to pay for the pass. The highest bidder will win the auction, but they will only pay **the second-highest bid**.

This is a **second-price sealed-bid auction**, often called a **Vickrey auction**.

Each fan has a private valuation for the backstage pass:

- Alex values it at £150 (they once wrote a fan algorithm about them).
- Casey values it at £100 (they're mainly here for the snacks).
- Jordan values it at £75 (they like the early albums).

Suppose the bids submitted are:

- Alex bids £150
- Casey bids £100
- Jordan bids £60

Then Alex wins, but only pays £100, the second-highest bid.

This setup has an interesting strategic property: **bidding your true value is a dominant strategy**. Even if Alex knew the other bids, they could not do better by bidding higher or lower than £150.

We'll explore why this is the case formally in what follows.

16.2 Theory

16.2.1 Definition: Auction Game

An **auction game** with N players (or bidders) is defined by:

- A set of random variables V_i, for $1 \leq i \leq N$, from which each player's

private valuation v_i for the good is drawn.

- A set of possible bids $b_i \in B_i$, where b_i is typically the output of a

bidding strategy $s_i : V_i \to B_i$ that maps valuations to bids.

- An **allocation rule**

$q : B_1 \times B_2 \times ... \times B_N \to [0,1]^N$,
which determines the probability with which each player receives the good.
Often, this output is a deterministic vector with a single 1 (winner) and the
remaining entries 0.

- A **payment rule**

$p : B_1 \times B_2 \times ... \times B_N \to \mathbb{R}^N$,
which determines how much each player pays as a function of all bids.

The utility of player i is then given by:

$$u_i = v_i \cdot q_i - p_i \tag{16.1}$$

where q_i is the allocation to player i, and p_i is their payment.

16.2.1.1 Example: The Auction Game for backstage passes

In Motivating Example: Bidding for backstage passes,
the auction game is defined with $N = 3$ players:

- The sampled valuations are: $v_1 = 150$, $v_2 = 100$, $v_3 = 75$.
- The chosen bids are: $b_1 = 150$, $b_2 = 100$, $b_3 = 60$.
- The allocation rule:

$$q_i(b) = \tag{16.2}$$

- The payment rule:

$$p_i(b) = \tag{16.3}$$

The resulting utilities for each player are:

- Alex: $u_1 = 150 \cdot 1 - 100 = 50$
- Casey: $u_2 = 100 \cdot 0 - 0 = 0$
- Jordan: $u_3 = 75 \cdot 0 - 0 = 0$

> **Note**
>
> For a different set of sampled valuations, the outcome could be entirely different. To study equilibrium behaviour in such settings, we will need new tools that can model beliefs and stochastic reasoning.

16.2.2 Definition: Bayesian Nash Equilibrium

A **Bayesian Nash equilibrium** is a strategy profile in a game with incomplete information such that, given their own type, each player's strategy maximises their expected utility assuming the strategies of the other players are fixed and that beliefs about types are correct.

16.2.3 Theorem: Bayesian Nash Equilibrium for Second Pay Auction

In a second price auction the Bayesian Nash equilibrium is for all players to bid their value:

$$b_i = v_i \tag{16.4}$$

Proof

Let us consider the strategy $b_i(v_i) = v_i$, that is, player i bids truthfully.

We show that this is a **best response** to all other players also bidding truthfully.

Fix a valuation v_i for player i. Let b_i denote the bid player i chooses (possibly different from v_i). Let $h = b_{(1)}^{(-i)}$ denote the highest bid among the other players. Since the game is a second-price auction, if player i wins they pay the highest of the remaining bids, which is exactly h. Player i's utility is therefore:

$$u_i = \tag{16.5}$$

Ties ($b_i = h$) occur with probability zero and are broken arbitrarily. Bidding $b_i = v_i$ gives utility $v_i - h > 0$ when $v_i > h$ and 0 when $v_i < h$. We compare this with the two possible deviations.

Case 1: $b_i < v_i$ (the deviation is to a lower bid than the value)

- If $h < b_i$, then $b_i > h$ still holds: player i still wins and still pays h, so the utility $v_i - h$ is unchanged.
- If $b_i < h < v_i$, then player i now loses an auction that truthful bidding would have won at a positive surplus $v_i - h > 0$, so the utility falls to 0.
- If $h > v_i$, then player i loses either way and the utility stays at 0.

Underbidding therefore never increases the utility.

Case 2: $b_i > v_i$ (the deviation is to a higher bid than the value)

- If $v_i > h$, then $b_i > h$ still holds: player i still wins and still pays h, so the utility $v_i - h$ is unchanged.
- If $b_i < h$, then player i loses either way and the utility stays at 0.
- If $v_i < h < b_i$, then player i now wins an auction that truthful bidding would have lost, paying $h > v_i$ for a negative surplus $v_i - h < 0$, so the utility falls below the 0 obtained by bidding truthfully.

Overbidding therefore never increases the utility.

In both cases, deviating from $b_i = v_i$ leaves the utility unchanged or decreases it, whatever the value of h. Hence bidding truthfully maximises utility for every v_i, and truthful bidding is a (weakly) dominant strategy.

Therefore, truthful bidding is a **Bayesian Nash equilibrium**.

16.2.4 Theorem: Bayesian Nash equilibrium in a first-price auction with uniform values

In a first-price auction with N players, where each valuation v_i is drawn independently from Unif$[0, 1]$, the Bayesian Nash equilibrium is for each player to **shade their bid** according to:

$$b_i = \frac{N-1}{N} v_i \tag{16.6}$$

Proof

Suppose all players follow the bidding strategy:

$$b_j = \frac{N-1}{N} v_j \quad \text{for all } j \neq i \tag{16.7}$$

Now consider a deviation by player i who bids $\bar{b} \neq \frac{N-1}{N} v_i$.

Let us compute the **expected utility** for player i from this deviation.

Player i wins if their bid $\bar{b}$ is higher than the bids of all other players, and receives utility equal to $v_i - \bar{b}$. If they lose, they get zero utility. Thus:

$$\mathbb{E}[u_i] = \Pr(\text{Win}) \cdot \left(v_i - \bar{b}\right) \tag{16.8}$$

Since all other players are bidding truthfully according to $b_j = \frac{N-1}{N} v_j$, and $v_j \sim \text{Unif}[0,1]$, we find:

$$\begin{aligned} \Pr(\text{Win}) &= \Pr\left(\bar{b} \geq \frac{N-1}{N} v_j \ \text{ for all } j \neq i\right) \\ &= \Pr\left(v_j \leq \frac{N}{N-1}\bar{b} \ \text{ for all } j \neq i\right) \\ &= \prod_{j \neq i} \Pr\left(v_j \leq \frac{N}{N-1}\bar{b}\right) \quad \text{(independence)} \\ &= \left(F\left(\frac{N}{N-1}\bar{b}\right)\right)^{N-1} \\ &= \left(\frac{N}{N-1}\bar{b}\right)^{N-1} \quad \text{(CDF of Unif}[0,1]\text{)} \end{aligned} \tag{16.9}$$

Hence, expected utility becomes:

$$\mathbb{E}[u_i] = \left(\frac{N}{N-1}\right)^{N-1} \bar{b}^{N-1}\left(v_i - \bar{b}\right) \tag{16.10}$$

To find the optimal deviation, we take the derivative of expected utility with respect to $\bar{b}$:

$$\begin{aligned} \frac{d\mathbb{E}[u_i]}{d\bar{b}} &= \left(\frac{N}{N-1}\right)^{N-1} \cdot \frac{d}{d\bar{b}}\left(\bar{b}^{N-1}\left(v_i - \bar{b}\right)\right) \\ &= \left(\frac{N}{N-1}\right)^{N-1} \left((N-1)\bar{b}^{N-2}\left(v_i - \bar{b}\right) - \bar{b}^{N-1}\right) \end{aligned} \tag{16.11}$$

Setting this derivative equal to zero gives a necessary condition for optimality:

$$\begin{aligned} (N-1)\bar{b}^{N-2}\left(v_i - \bar{b}\right) - \bar{b}^{N-1} &= 0 \\ (N-1)\left(v_i - \bar{b}\right) - \bar{b} &= 0 \quad \text{(divide by } \bar{b}^{N-2} \neq 0\text{)} \\ (N-1)v_i - N\bar{b} &= 0 \\ \bar{b} &= \frac{N-1}{N} v_i \end{aligned} \tag{16.12}$$

It remains to confirm that this stationary point is the global maximum. Player i never bids above v_i, since winning at a price $\bar{b} > v_i$ gives a negative surplus $v_i - \bar{b} < 0$, and never below 0, so we may restrict to $\bar{b} \in [0, v_i]$. On this interval $\bar{b} < \frac{N-1}{N} < 1$, so the argument $\frac{N}{N-1}\bar{b}$ of the uniform CDF stays in $[0,1]$ and the expression for $\mathbb{E}[u_i]$ above is valid throughout. The expected utility $\left(\frac{N}{N-1}\right)^{N-1}\bar{b}^{N-1}\left(v_i - \bar{b}\right)$ is zero at both endpoints $\bar{b} = 0$ and $\bar{b} = v_i$ and strictly positive in between, so its unique interior stationary point $\bar{b} = \frac{N-1}{N} v_i$ is the global maximum on $[0, v_i]$.

Thus, any deviation $\bar{b} \neq \frac{N-1}{N} v_i$ does not improve utility, and the symmetric profile in which every player bids $\frac{N-1}{N} v_i$ is a Bayesian Nash equilibrium.

16.3 Exercises

Exercise 16.1:

In a second-price auction with two players whose private valuations are:

- Player 1: $v_1 = 0.8$
- Player 2: $v_2 = 0.6$

Each player submits a sealed bid. Show that bidding truthfully is a dominant strategy for each player, and compute the resulting allocation, payment, and utility.

Exercise 16.2:

Consider a second-price auction with $v_1 = 0.9$, $v_2 = 0.7$, and $v_3 = 0.5$.

Suppose Player 1 deviates and bids $b_1 = 1.0$, Player 2 bids truthfully, and Player 3 bids $b_3 = 0.6$.

- Who wins the auction?
- What is the price paid?
- What is Player 1's utility?
- Would Player 1 have been better off bidding truthfully?

Exercise 16.3:

Let $N = 2$ players, each with valuation $v_i \sim \text{Unif}[0, 1]$, use the symmetric strategy $b_i = \frac{1}{2} v_i$.

Compute the expected utility of Player 1 as a function of their value v_1.

Hint: integrate over the opponent's valuation or bid.

Exercise 16.4:

Using the first-price auction model with $v_i \sim \text{Unif}[0, 1]$ and the assumption that all other players bid $b_j = \alpha v_j$, show that the best response of Player i is to bid $b_i = \frac{N-1}{N} v_i$ when $\alpha = \frac{N-1}{N}$.

Exercise 16.5:

Suppose players' values are drawn from an exponential distribution with CDF $F(v) = 1 - e^{-\lambda v}$, and each player bids $b_i = \beta v_i$ in a first-price auction.

- Derive the probability of winning for a given bid $\bar{b}$.
- Derive the expected utility and determine the value of β

that maximises expected utility for $N = 2$.

Challenge: compare the optimal β to the uniform case.

Exercise 16.6:

Suppose there are N symmetric bidders with valuations $v_i \sim \text{Unif}[0, 1]$, drawn independently. In both the first-price and second-price sealed-bid auctions, assume all players follow the Bayesian Nash equilibrium bidding strategies:

- First-price: $b_i = \frac{N-1}{N} v_i$
- Second-price: $b_i = v_i$

Let R denote the expected revenue of the seller.

Task: Show that both mechanisms yield the same expected revenue:

$$\mathbb{E}[R_{\text{first}}] = \mathbb{E}[R_{\text{second}}] \tag{16.13}$$

Hint:
In $\text{Unif}[0, 1]$, the expected value of the k-th order statistic (i.e., the k-th smallest value) is:

$$\mathbb{E}\left[v_{(k)}\right] = \frac{k}{N+1} \tag{16.14}$$

16.4 Programming

There is a python library called `sold` which allows for the numeric simulation of auctions. The code below builds the valuation distributions and bidding rules for an auction with $N = 3$ players where the first two players bid the true value and the third player uses the strategy of Theorem: Bayesian Nash equilibrium in a first-price auction with uniform values.

```
import scipy.stats
import numpy as np

import sold
import sold.allocate
import sold.bid
import sold.pay

N = 3
valuation_distributions = [scipy.stats.uniform() for _ in range(N)]
bidding_functions = [sold.bid.true_value for _ in range(N - 1)] + [
```

```
    sold.bid.create_shaded_bid_map((N - 1) / N)
]
```

Now let us simulate this 100 times to confirm that the shaded big strategy gets a higher utility:

```
repetitions = 100
utilities = []
for seed in range(repetitions):
    allocation, payments, valuations = sold.auction(
        valuation_distributions=valuation_distributions,
        bidding_functions=bidding_functions,
        allocation_rule=sold.allocate.first_price,
        payment_rule=sold.pay.first_price,
        seed=seed,
    )
    utilities.append(valuations - allocation * payments)
mean_utilities = np.mean(utilities, axis=0)
print(f"Mean utility per bidder: {mean_utilities}")
```

```
Mean utility per bidder: [0.17535816 0.18842919 0.47175621]
```

16.5 Notable Research

One of the early important publications in this field is (Vickrey, 1961), which not only introduced the second-price auction, often called the Vickrey auction, proving that truth-telling is optimal, but also discusses the first-price auction. Vickrey was awarded the Nobel Prize in Economics in 1996 for this foundational work.

Another foundational contribution is (Myerson, 1981), which formulates the design of optimal auctions, introduces the concept of virtual valuations, and proves a generalisation of the result in Exercise 6: the revenue equivalence theorem. This theorem shows that all standard auctions yield the same expected revenue when bidders are risk-neutral and have independent private values.

Auction theory is often described as one of game theory's most successfully applied branches. A striking example is the role of auctions in modern online advertising. This is exemplified in the work of Google's chief economist (Varian, 2007), which provides a theoretical model of position auctions, the type of auctions used to allocate advertising slots in search engines.

Further Nobel-recognised contributions to auction theory come from Paul Milgrom and Robert Wilson, who were awarded the Nobel Prize in Economics in 2020. Their work extends auction theory to settings where bidders' values are interdependent and shaped by shared information. Wilson's early work (Wilson, 1967) developed models of bidding under common values, where bidders must reason about both their own and others' information. Milgrom built on this by analysing how different auction formats perform in such settings (Milgrom & Weber, 1982), helping to guide the design of real-world mechanisms. Together, they co-designed the simultaneous multiple-round auction (SMRA) used to allocate radio spectrum, which has had significant practical impact.

16.6 Conclusion

Auction theory provides a rigorous framework for understanding how goods and resources can be allocated efficiently under conditions of incomplete information. We've seen that second-price auctions promote truthful bidding as a dominant strategy, while first-price auctions require bidders to strategically shade their bids. The concept of Bayesian Nash equilibrium allows us to analyse these

strategies under uncertainty, and the revenue equivalence theorem links together many of the most commonly used auction formats.

These results underpin some of the most economically significant allocation mechanisms in use today, from government spectrum sales to online advertising platforms.

Table 16.1 gives a summary of the concepts seen in this chapter as well as a list of other auction types.

Auction Format	Bidding Strategy	Truthful?	Allocates to Highest Bidder?	Typical Use Case
Second-price sealed	Bid your true value	✅	✅	eBay-style auctions
First-price sealed	Shade your bid	❌	✅	Procurement, real estate
English (ascending)	Stay in until your value	✅*	✅	Art, livestock auctions
Dutch (descending)	Accept before price drops	❌	✅	Cut flowers, perishable goods
Position auction	Rank-based bidding	❌	✅ (under assumptions)	Online advertising (e.g., Google Ads)
All-pay auction	Bid your value (often)	❌	✅	Tournaments, lobbying, rent-seeking

Table 16.2: *Truthful under private values and risk neutrality.
Table 16.1: Different types of auctions and their properties

Important

Auction theory assumes that players form beliefs about other players' valuations. To analyse strategic behaviour under uncertainty, we use the concept of **Bayesian Nash equilibrium**, an equilibrium defined in expectation. This framework allows auctions to be designed in ways that incentivise truthful bidding.

16.7 Solutions

Solution 16.1: Solution to Exercise 1

We have two players with valuations $v_1 = 0.8$ and $v_2 = 0.6$.

Truthful bidding is a dominant strategy for each player.

We show this for Player 1; the argument for Player 2 is identical by symmetry.

Fix an arbitrary bid b_2 from Player 2. Player 1 chooses bid b_1.

In a second-price auction, Player 1's utility is:

$$u_1 = \tag{16.15}$$

(Ties are broken arbitrarily; they occur on a set of measure zero.)

We consider two cases:

Case 1: $b_2 < v_1 = 0.8$.

- If $b_1 = v_1 = 0.8 > b_2$: Player 1 wins and receives $u_1 = v_1 - b_2 > 0$.
- If $b_1 < b_2$: Player 1 loses and receives $u_1 = 0$.
- If $b_1 > b_2$ (but $b_1 \neq v_1$): Player 1 still wins and still pays b_2. Utility is the same $v_1 - b_2 > 0$.

Bidding v_1 guarantees winning (and the positive surplus) whenever it is profitable to do so. No deviation improves utility.

Case 2: $b_2 > v_1 = 0.8$.

- If $b_1 = v_1 = 0.8 < b_2$: Player 1 loses and receives $u_1 = 0$.
- If $b_1 > b_2$: Player 1 wins but pays $b_2 > v_1$, giving $u_1 = v_1 - b_2 < 0$.

Bidding v_1 correctly results in losing (utility 0), which is better than winning at a loss. No deviation to a lower bid changes the outcome.

In all cases, deviating from $b_1 = v_1$ either leaves utility unchanged or makes it worse. Hence $b_i = v_i$ is a **dominant strategy** for each player.

Resulting allocation, payment, and utilities (with truthful bids):

Both players bid their true values: $b_1 = 0.8$, $b_2 = 0.6$.

- Player 1 has the highest bid, so Player 1 **wins**.
- Payment: Player 1 pays the second-highest bid, which is $b_2 = 0.6$.
- Utilities:

$$u_1 = v_1 - b_2 = 0.8 - 0.6 = 0.2, \qquad u_2 = 0. \tag{16.16}$$

```
v1, v2 = 0.8, 0.6
b1, b2 = v1, v2  # truthful bids

winner = 1 if b1 > b2 else 2
price = b2 if winner == 1 else b1  # second-highest bid

u1 = (v1 - price) if winner == 1 else 0
u2 = (v2 - price) if winner == 2 else 0

print(f"Winner: Player {winner}")
print(f"Price paid: {price}")
print(f"Utility of Player 1: {u1}")
print(f"Utility of Player 2: {u2}")
```

```
Winner: Player 1
Price paid: 0.6
Utility of Player 1: 0.20000000000000007
Utility of Player 2: 0
```

Solution 16.2: Solution to Exercise 2

We have valuations $v_1 = 0.9$, $v_2 = 0.7$, $v_3 = 0.5$.

The bids submitted are $b_1 = 1.0$, $b_2 = 0.7$ (truthful), $b_3 = 0.6$.

Who wins the auction?

The highest bid is $b_1 = 1.0$, so **Player 1 wins**.

What is the price paid?

In a second-price auction the winner pays the second-highest bid:

$$\text{price} = \max(b_2, b_3) = \max(0.7, 0.6) = 0.7. \tag{16.17}$$

What is Player 1's utility?

$$u_1 = v_1 - \text{price} = 0.9 - 0.7 = 0.2. \tag{16.18}$$

Would Player 1 have been better off bidding truthfully?

If Player 1 bids $b_1 = v_1 = 0.9$ instead:

- $b_1 = 0.9 > b_2 = 0.7 > b_3 = 0.6$, so Player 1 still wins.
- Price paid: $\max(0.7, 0.6) = 0.7$ (unchanged, since it is determined by the other bids).
- Utility: $u_1 = 0.9 - 0.7 = 0.2$ (the same).

Player 1 would **not** have been better or worse off by bidding truthfully. This illustrates the key property of second-price auctions: the winner's payment depends only on others' bids, so overbidding above your value (when you already win) does not affect the price you pay. Truthful bidding is weakly dominant.

```
v1, v2, v3 = 0.9, 0.7, 0.5
b1_deviate, b2, b3 = 1.0, 0.7, 0.6
b1_truthful = v1

def second_price_outcome(bids, valuations):
    winner = bids.index(max(bids))
    sorted_bids = sorted(bids, reverse=True)
    price = sorted_bids[1]
    utils = [v - price if i == winner else 0 for i, v in
enumerate(valuations)]
    return winner + 1, price, utils

winner_d, price_d, utils_d = second_price_outcome(
    [b1_deviate, b2, b3], [v1, v2, v3]
)
winner_t, price_t, utils_t = second_price_outcome(
    [b1_truthful, b2, b3], [v1, v2, v3]
)

print("With overbid b1=1.0:")
print(f"  Winner: Player {winner_d}, Price: {price_d}, u1={utils_d[0]}")
print("With truthful bid b1=0.9:")
print(f"  Winner: Player {winner_t}, Price: {price_t}, u1={utils_t[0]}")
```

```
With overbid b1=1.0:
  Winner: Player 1, Price: 0.7, u1=0.20000000000000007
With truthful bid b1=0.9:
  Winner: Player 1, Price: 0.7, u1=0.20000000000000007
```

Solution 16.3: Solution to Exercise 3

We have $N = 2$ players with $v_i \sim \text{Unif}[0, 1]$ using the symmetric strategy $b_i = \frac{1}{2}v_i$.

Expected utility of Player 1 as a function of v_1:

Player 1 bids $b_1 = \frac{1}{2}v_1$. Player 2 bids $b_2 = \frac{1}{2}v_2$.

Player 1 wins if $b_1 > b_2$, i.e., $\frac{1}{2}v_1 > \frac{1}{2}v_2$, i.e., $v_2 < v_1$.

When Player 1 wins, they pay their own bid $b_1 = \frac{1}{2}v_1$ (first-price auction). When Player 1 loses, they receive utility 0.

$$\mathbb{E}[u_1 \texttt{ mid } v_1] = \Pr(v_2 < v_1) \cdot (v_1 - b_1) + \Pr(v_2 \geq v_1) \cdot 0 \tag{16.19}$$

Since $v_2 \sim \text{Unif}[0, 1]$:

$$\Pr(v_2 < v_1) = v_1 \tag{16.20}$$

Therefore:

$$\mathbb{E}[u_1 \texttt{ mid } v_1] = v_1 \cdot \left(v_1 - \frac{1}{2}v_1\right) = v_1 \cdot \frac{v_1}{2} = \frac{v_1^2}{2} \tag{16.21}$$

This can also be verified by integrating over v_2:

$$\begin{aligned} &\mathbb{E}[u_1 \texttt{ mid } v_1] \\ &\qquad = \int_0^{v_1} \left(v_1 - \frac{1}{2}v_1\right) dv_2 \\ &\qquad = \int_0^{v_1} \frac{v_1}{2}\, dv_2 \\ &\qquad = \frac{v_1}{2} \cdot v_1 \\ &\qquad = \frac{v_1^2}{2}. \end{aligned} \tag{16.22}$$

So the expected utility of Player 1 is:

$$\mathbb{E}[u_1 \texttt{ mid } v_1] = \frac{v_1^2}{2} \tag{16.23}$$

```python
import sympy as sym

v1, v2 = sym.symbols("v_1 v_2", positive=True)

b1 = sym.Rational(1, 2) * v1
b2 = sym.Rational(1, 2) * v2

# Utility when v2 < v1 (Player 1 wins): v1 - b1
utility_win = v1 - b1

# Integrate over v2 from 0 to v1
expected_utility = sym.integrate(utility_win, (v2, 0, v1))
sym.simplify(expected_utility)
```

```
v_1**2/2
```

Solution 16.4: Solution to Exercise 4

We have N players with $v_i \sim \text{Unif}[0,1]$, and all other players bid $b_j = \alpha v_j$ for some $\alpha > 0$. We derive the best response of Player i.

Setting up the expected utility:

Player i bids $\bar{b}$. Player i wins if $\bar{b} > b_j = \alpha v_j$ for all $j \neq i$, i.e., if $v_j < \bar{b}/\alpha$ for all $j \neq i$.

Since $v_j \sim \text{Unif}[0,1]$ independently:

$$\Pr(\text{Win}) = \prod_{j \neq i} \Pr\left(v_j < \frac{\bar{b}}{\alpha}\right) = \left(\frac{\bar{b}}{\alpha}\right)^{N-1} \tag{16.24}$$

(assuming $\bar{b}/\alpha \leq 1$, i.e., $\bar{b} \leq \alpha$).

In a first-price auction, the winner pays their own bid, so:

$$\mathbb{E}[u_i \texttt{ mid } v_i] = \left(\frac{\bar{b}}{\alpha}\right)^{N-1} \left(v_i - \bar{b}\right) \tag{16.25}$$

Optimising over $\bar{b}$:

$$\begin{aligned} &\frac{d}{d\bar{b}} \mathbb{E}[u_i] \\ &\quad = \frac{1}{\alpha^{N-1}} \frac{d}{d\bar{b}} \left[\bar{b}^{N-1}\left(v_i - \bar{b}\right)\right] \\ &\quad = \frac{1}{\alpha^{N-1}} \left[(N-1)\bar{b}^{N-2}\left(v_i - \bar{b}\right) - \bar{b}^{N-1}\right] \end{aligned} \tag{16.26}$$

Setting this equal to zero and dividing by $\bar{b}^{N-2} \neq 0$:

$$(N-1)\left(v_i - \bar{b}\right) - \bar{b} = 0 \tag{16.27}$$

$$(N-1)v_i = N\bar{b} \tag{16.28}$$

$$\bar{b}^* = \frac{N-1}{N} v_i \tag{16.29}$$

Confirming $\alpha = \frac{N-1}{N}$ yields a symmetric equilibrium:

When $\alpha = \frac{N-1}{N}$, the best response of Player i is to bid $\frac{N-1}{N} v_i$, which equals αv_i. Therefore, the strategy $b_i = \frac{N-1}{N} v_i$ is a best response to all other players using the same strategy, confirming it is a **Bayesian Nash equilibrium**.

The second-order condition confirms this is a maximum: the expected utility as a function of $\bar{b}$ is a degree-N polynomial in $\bar{b}$ with a negative leading coefficient, so the unique interior critical point is indeed a maximum.

```
import sympy as sym

b_bar, v_i, alpha = sym.symbols(r"\bar{b} v_i alpha", positive=True)
N = sym.Symbol("N", positive=True, integer=True)

# Expected utility
```

```
EU = (b_bar / alpha) ** (N - 1) * (v_i - b_bar)

# First order condition
dEU = sym.diff(EU, b_bar)
best_response = sym.solve(dEU, b_bar)
print("Best response:", best_response)
```

```
Best response: [v_i - v_i/N]
```

Solution 16.5: Solution to Exercise 5

Players' valuations are drawn from an exponential distribution with CDF $F(v) = 1 - e^{-\lambda v}$, and each player bids $b_i = \beta v_i$ in a first-price auction with N players. We focus on $N = 2$.

1. **Probability of winning for a given bid $\bar{b}$:**

Suppose all other players bid $b_j = \beta v_j$. Player i wins if $\bar{b} > \beta v_j$ for all $j \neq i$, i.e., $v_j < \bar{b}/\beta$ for all $j \neq i$.

$$\Pr(\text{Win}) = \prod_{j \neq i} F\left(\frac{\bar{b}}{\beta}\right) = \left(1 - e^{-\lambda \bar{b}/\beta}\right)^{N-1} \tag{16.30}$$

For $N = 2$:

$$\Pr(\text{Win}) = 1 - e^{-\lambda \bar{b}/\beta} \tag{16.31}$$

2. **Expected utility and optimal β:**

For $N = 2$, the expected utility of Player i bidding $\bar{b}$ (first-price auction, paying their own bid when they win) is:

$$\mathbb{E}[u_i \texttt{ mid } v_i] = \left(1 - e^{-\lambda \bar{b}/\beta}\right)\left(v_i - \bar{b}\right) \tag{16.32}$$

Taking the derivative with respect to $\bar{b}$ and setting it to zero:

$$\frac{d}{d\bar{b}}\mathbb{E}[u_i] = \frac{\lambda}{\beta} e^{-\lambda \bar{b}/\beta}\left(v_i - \bar{b}\right) - \left(1 - e^{-\lambda \bar{b}/\beta}\right) = 0 \tag{16.33}$$

Let $\mu = \lambda \bar{b}/\beta$ for brevity. Then $\bar{b} = \mu\beta/\lambda$ and:

$$\begin{aligned} \mu e^{-\left(\mu v_i - \frac{\mu\beta}{\lambda}\right)} \\ = \frac{\beta}{\lambda}(1 - e^{-\mu}) \end{aligned} \tag{16.34}$$

For a symmetric equilibrium we require $\bar{b} = \beta v_i$, i.e., the best response $\bar{b}^*$ must equal βv_i. Substituting $\bar{b} = \beta v_i$:

$$\frac{\lambda}{\beta} e^{-\lambda v_i}(v_i - \beta v_i) = 1 - e^{-\lambda v_i} \tag{16.35}$$

$$\lambda v_i (1 - \beta) e^{-\lambda v_i} = \frac{\beta}{\lambda} \cdot \frac{\lambda}{\beta}\left(1 - e^{-\lambda v_i}\right) \tag{16.36}$$

More carefully, substituting $\bar{b} = \beta v_i$ into the first-order condition:

$$\frac{\lambda}{\beta} e^{-\lambda v_i} \cdot v_i(1-\beta) = 1 - e^{-\lambda v_i} \tag{16.37}$$

This must hold for all $v_i > 0$, which means β is determined implicitly. For the exponential distribution, unlike the uniform case, the linear bidding strategy $b = \beta v$ does not generally yield a closed-form symmetric equilibrium in terms of β alone. Instead, the equilibrium bid function is derived via the general first-order approach:

$$b^*(v) = \frac{\int_0^v t\, f(t)\, F(t)^{N-2} dt}{F(v)^{N-1}} \quad (\text{for } N \geq 2\ ) \tag{16.38}$$

For $N = 2$ and $F(v) = 1 - e^{-\lambda v}$, $f(v) = \lambda e^{-\lambda v}$:

$$\begin{aligned} & b^*(v) \\ & = \frac{\int_0^v t \cdot \lambda e^{-\lambda t} dt}{1 - e^{-\lambda v}} \end{aligned} \tag{16.39}$$

Integrating by parts: $\int_0^v t\lambda e^{-\lambda t}\, dt = \left[-te^{-\lambda t}\right]_0^v + \int_0^v e^{-\lambda t}\, dt = -ve^{-\lambda v} + \frac{1-e^{-\lambda v}}{\lambda}$

So:

$$\begin{aligned} b^*(v) &= \frac{-ve^{-\lambda v} + \left(1 - e^{-\lambda v}\right)/\lambda}{1 - e^{-\lambda v}} \\ &= \frac{1}{\lambda} - \frac{ve^{-\lambda v}}{1 - e^{-\lambda v}} \end{aligned} \tag{16.40}$$

Comparison with the uniform case: For Unif$[0, 1]$ with $N = 2$, the equilibrium bid is $b^*(v) = v/2$, which is a linear fraction of v. For the exponential distribution the equilibrium bid function $b^*(v)$ is nonlinear in v and is always less than v (the bidder shades downward), but the degree of shading depends on v in a nonlinear way.

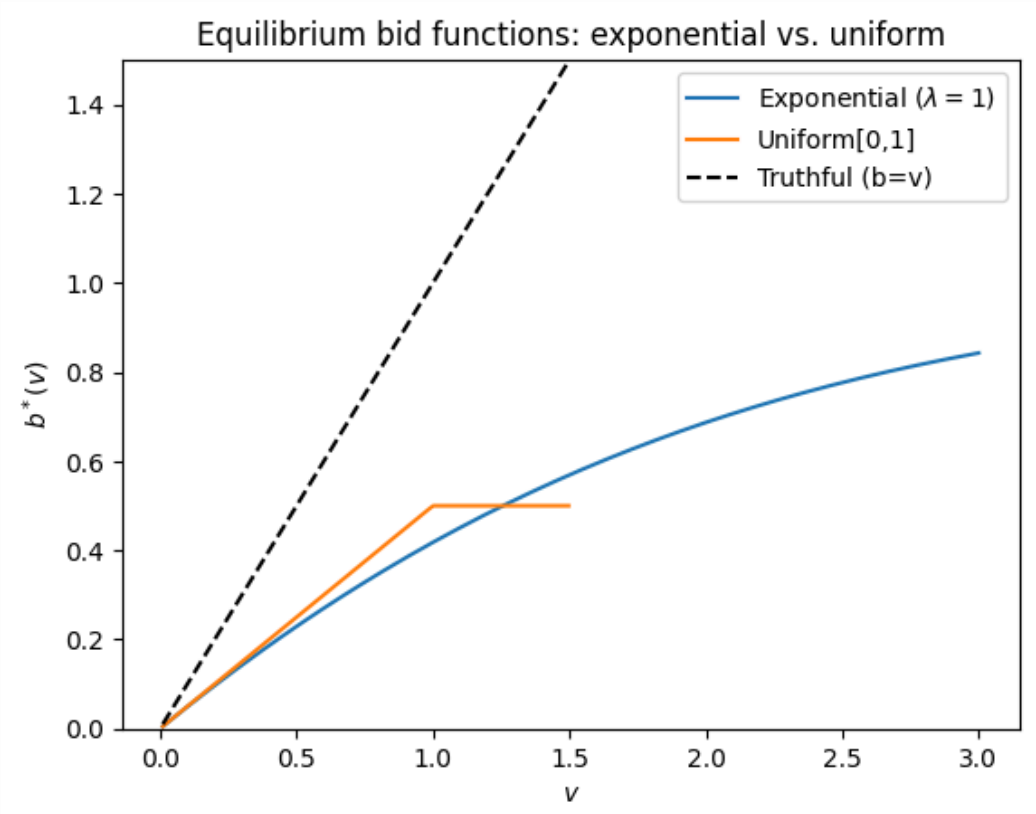
```
import sympy as sym

v, lam = sym.symbols("v lambda", positive=True)

f_v = lam * sym.exp(-lam * v)
F_v = 1 - sym.exp(-lam * v)

# N=2: equilibrium bid function
numerator = sym.integrate(v * f_v, (v, 0, v))
b_star = sym.simplify(numerator / F_v)
print("Equilibrium bid function (N=2, exponential):")
b_star
```

```
Equilibrium bid function (N=2, exponential):
```

```
-(lambda*v - exp(lambda*v) + 1)/(lambda*(exp(lambda*v) - 1))
```

```
# Compare to uniform case: b*(v) = v/2
import numpy as np
import matplotlib.pyplot as plt

lam_val = 1.0
```

```
v_vals = np.linspace(0.01, 3, 200)
b_exp = 1/lam_val - v_vals * np.exp(-lam_val * v_vals) / (1 - np.exp(-lam_val
* v_vals))
b_uniform = 0.5 * np.clip(v_vals, 0, 1)

plt.figure()
plt.plot(v_vals, b_exp, label=r"Exponential ($\lambda=1$)")
plt.plot(v_vals[:100], b_uniform[:100], label="Uniform[0,1]")
plt.plot(v_vals, v_vals, "k--", label="Truthful (b=v)")
plt.xlabel("$v$")
plt.ylabel("$b^*(v)$")
plt.title("Equilibrium bid functions: exponential vs. uniform")
plt.legend()
plt.ylim(0, 1.5)
plt.show()
```



Solution 16.6: Solution to Exercise 6

We have N symmetric bidders with $v_i \sim \text{Unif}[0, 1]$ independently.

Expected revenue of the first-price auction:

At equilibrium, each bidder bids $b_i = \frac{N-1}{N} v_i$. The seller's revenue equals the highest bid:

$$
\begin{aligned}
R_{\text{first}} &= \max_i \left(\frac{N-1}{N} v_i \right) \\
&= \frac{N-1}{N} \max_i (v_i) \\
&= \frac{N-1}{N} v_{(N)}
\end{aligned}
\tag{16.41}
$$

where $v_{(N)}$ denotes the highest order statistic (maximum) of N i.i.d. Unif$[0, 1]$ random variables. Using the hint:

$$
\mathbb{E}\left[v_{(N)}\right] = \frac{N}{N+1}
\tag{16.42}
$$

Therefore:

$$
\mathbb{E}[R_{\text{first}}] = \frac{N-1}{N} \cdot \frac{N}{N+1} = \frac{N-1}{N+1}
\tag{16.43}
$$

Expected revenue of the second-price auction:

At equilibrium, each bidder bids truthfully $b_i = v_i$. The winner (highest bidder) pays the second-highest bid $v_{(N-1)}$:

$$
R_{\text{second}} = v_{(N-1)}
\tag{16.44}
$$

where $v_{(N-1)}$ is the second-highest order statistic. Using the hint:

$$
\mathbb{E}\left[v_{(N-1)}\right] = \frac{N-1}{N+1}
\tag{16.45}
$$

Therefore:

$$
\mathbb{E}[R_{\text{second}}] = \frac{N-1}{N+1}
\tag{16.46}
$$

Conclusion:

$$
\mathbb{E}[R_{\text{first}}] = \frac{N-1}{N+1} = \mathbb{E}[R_{\text{second}}]
\tag{16.47}
$$

Both mechanisms yield the same expected revenue. This is the **Revenue Equivalence Theorem** in the special case of uniform valuations.

The result holds because both auctions are efficient (the highest-value bidder wins) and both give zero utility to the lowest type ($v_i = 0$ receives 0). By the Revenue Equivalence Theorem, any two auctions with these two properties yield the same expected revenue for the seller.

```python
import sympy as sym

N = sym.Symbol("N", positive=True, integer=True)

# Expected value of k-th order statistic of N Unif[0,1] r.v.'s is k/(N+1)
# First-price expected revenue
E_v_N = N / (N + 1)           # E[v_{(N)}]
E_R_first = (N - 1) / N * E_v_N
E_R_first_simplified = sym.simplify(E_R_first)
print("E[R_first] =", E_R_first_simplified)
```

```python
# Second-price expected revenue
E_v_N1 = (N - 1) / (N + 1)    # E[v_{(N-1)}]
E_R_second = E_v_N1
print("E[R_second] =", sym.simplify(E_R_second))

print("Equal?", sym.simplify(E_R_first_simplified - E_R_second) == 0)
```

```
E[R_first] = (N - 1)/(N + 1)
E[R_second] = (N - 1)/(N + 1)
Equal? True
```

```python
import numpy as np

# Numerical verification for N=3
np.random.seed(42)
n_sims = 200_000
N_val = 3

valuations = np.random.uniform(0, 1, (n_sims, N_val))

# First-price: each bid (N-1)/N * v, revenue = max bid
bids_fp = (N_val - 1) / N_val * valuations
revenue_fp = bids_fp.max(axis=1)

# Second-price: bid truthfully, revenue = second-highest bid
sorted_vals = np.sort(valuations, axis=1)
revenue_sp = sorted_vals[:, -2]

print(f"N={N_val}")
print(f"Expected revenue (first-price):  {revenue_fp.mean():.4f}")
print(f"Expected revenue (second-price): {revenue_sp.mean():.4f}")
print(f"Theoretical value (N-1)/(N+1):  {(N_val-1)/(N_val+1):.4f}")
```

```
N=3
Expected revenue (first-price):  0.5001
Expected revenue (second-price): 0.5003
Theoretical value (N-1)/(N+1):  0.5000
```

17. Social Choice

Combining the preferences of many individuals into a single collective decision is far harder than it appears. This chapter studies social choice theory, characterising aggregation procedures and establishing impossibility results, most famously Arrow's theorem, that reveal deep constraints on what any fair voting rule can achieve.

17.1 Motivating Example: Voting on exam topics

In the final week of term, a game theory class of 45 students receives an unexpected offer from their professor:

> "You get to vote on which topic will feature most heavily on the final exam. Each of you will submit a full ranking of the three options, and we'll use pairwise majority voting to decide the winner."

The three candidate topics are:

- **A. Replicator dynamics**
- **B. Best response polytopes**
- **C. Repeated games**

After collecting all the votes, the professor finds that the students fall into the preference groups shown in Table 17.1.

Group size	1st choice	2nd choice	3rd choice
18 students	A	B	C
16 students	B	C	A
11 students	C	A	B

If the professor only looks at the first choice then Replicator dynamics will be the chosen topic of the exam. However let us examine each pairwise contest:

- **A vs B**
  - 18 students: prefer **A to B** and B to C
  - 16 students: prefer B to C and C to A so prefer **B to A**
  - 11 students: prefer C to A and **A to B**.

So **A beats B**, with 29 votes to 16.

- **B vs C**
  - 18 students: prefer A to B and **B to C**
  - 16 students: prefer **B to C** and C to A
  - 11 students: prefer C to A and A to B so prefer **C to B**

So **B beats C**, with 34 votes to 11.

- **C vs A**
  - 18 students: prefer A to B and B to C so prefer **A to C**
  - 16 students: prefer B to C and **C to A**
  - 11 students: prefer **C to A** and A to B arrow.r **C beats A**, with 27 votes to 18.

A majority prefers A to B, B to C, and C to A.

This cycle is called a **Condorcet cycle** (Caritat, 1785) and it illustrates the core difficulty in aggregating preferences from a group: **majority rule may not produce a consistent collective ranking**. This motivates a central question in voting theory:

> What properties should a "reasonable" voting rule satisfy, and can we always satisfy them all at once?

== Theory

17.1.1 Definition: Preference Function

Let X be a finite set of alternatives. A **preference function** for a voter is a strict linear order $\succ$ on X, meaning that for all distinct alternatives $x, y, z \in X$:

- **Asymmetry**: if $x \succ y$, then not $y \succ x$
- **Transitivity**: if $x \succ y$ and $y \succ z$, then $x \succ z$
- **Completeness**: for all $x \neq y$, either $x \succ y$ or $y \succ x$

We refer to a complete list of n voters' preference functions as a **preference profile**.

17.1.1.1 Example: Preference Function for the Exam Vote

For Motivating Example: Voting on exam topics the set of alternative is:

$$A = \{A, B, C\} \tag{17.1}$$

The preference profile is:

- Group 1: $A \succ B \succ C$
- Group 2: $B \succ C \succ A$
- Group 3: $C \succ A \succ B$

17.1.2 Definition: A Social Welfare Function

A **social welfare function** is a rule that maps every possible preference profile over a set of alternatives X to a strict linear order on X.

That is, it takes individual rankings and produces a **collective ranking** of all the alternatives.

17.1.2.1 Example: Social Welfare Function for the Exam Vote

For Motivating Example: Voting on exam topics one social welfare function would be to rank the outcomes based on number of first choice votes, giving:

$$(A, B, C) \tag{17.2}$$

17.1.3 Definition: Simple Majority Rule

A **simple majority rule** is a voting rule that compares alternatives x and y based on how many voters prefer x over y:

- x is ranked above y in the collective outcome if and only if a **strict majority** of voters prefer x to y.

This defines a binary relation over pairs, but may not yield a transitive overall ranking. When it does, the top-ranked option is called the **Condorcet winner**.

17.1.4 Theorem: Arrow's Impossibility Theorem

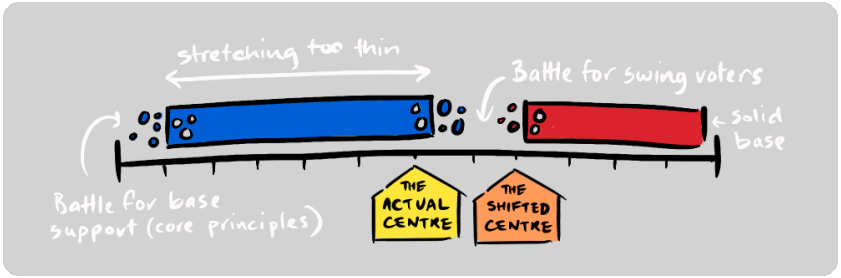


Figure 17.1: Where the collective 'centre' sits depends on how individual views are weighed against one another. Arrow's theorem shows that no aggregation rule can reconcile a few reasonable fairness requirements all at once.

The following theorem is given without proof (Arrow, 1951).

Let X be a set of at alternatives with $\mid X \mid \geq 3$. There exists no social welfare function that satisfies **all** of the following properties:

1. **Unrestricted domain**: all possible preference profiles are allowed
2. **Pareto efficiency**: if all voters prefer x over y, then x is ranked above y
3. **Independence of irrelevant alternatives (IIA)**: the group's relative ranking of x and y depends only on how individuals rank x vs y
4. **Non-dictatorship**: there is no single voter whose preferences always determine the outcome

In short, there is **no perfect voting rule** that aggregates individual preferences into a consistent group ranking while satisfying all of these fairness criteria.

In the context of Motivating Example: Voting on exam topics this theorem implies that there is no perfect way to choose. However there are two approaches that can offer a good way to obtain an outcome.

17.1.5 Definition: Condorcet's Method

An alternative x is a **Condorcet winner** if, for every other alternative y, a strict majority of voters prefer x over y.

> **Note**
>
> A Condorcet winner does not always exist due to the possibility of **majority cycles**, as illustrated in the motivating example.

In the context of Motivating Example: Voting on exam topics we have already seen that no Condorcet winner exists.

17.1.6 Definition: Borda's Method

Let x be a finite set of alternatives. Every alternative obtains k points from a voter if that voter ranks the alternative above exactly k other alternatives. The total Borda score of each alternative is the sum of its points across all voters. The alternative with the highest total score is the **Borda winner**.

Borda's method always produces a complete ranking, but it may fail to select a Condorcet winner when one exists.

Important

The Borda and Condorcet methods do not escape Arrow's theorem, both violate independence of irrelevant alternatives, but they do represent well-motivated responses to it. By giving up IIA, they retain the other desirable properties of fairness, including Pareto efficiency and non-dictatorship. In practice, they are widely regarded as reasonable voting rules, especially when full transitive rankings are available.

17.1.6.1 Example: The Borda Method for the Exam Vote

For the Motivating Example: Voting on exam topics:

- the 18 individuals in the first group give 2 points to option A and 1 point to option B.
- the 16 individuals in the second group give 2 points to option B and 1 point to option C.
- the 11 individuals in the third group give 2 points to option C and 1 point to option A.

Thus the Borda score of each option is:

- A: $18 \cdot 2 + 11 \cdot 1 = 47$
- B: $18 \cdot 1 + 16 \cdot 2 = 50$
- C: $11 \cdot 2 + 16 \cdot 1 = 38$

The Borda winner is thus B.

17.2 Exercises

Exercise 17.1:

Consider the following preference profile over three alternatives $X = \{A, B, C\}$:

- 3 voters: $A \succ B \succ C$
- 2 voters: $B \succ C \succ A$
- 2 voters: $C \succ A \succ B$

a. Construct the pairwise majority contests.
b. Is there a Condorcet winner?
c. Is the majority preference relation transitive?

Exercise 17.2:

Consider the preference profile:

- 4 voters: $A \succ B \succ C$
- 3 voters: $B \succ C \succ A$
- 2 voters: $C \succ A \succ B$

a. Compute the Borda scores of each alternative.
b. Determine if a Condorcet winner exists.
c. Do the Borda and Condorcet methods select the same winner?

Exercise 17.3:

Suppose an election uses the Borda count and the following preference profile:

- 5 voters: $A \succ B \succ C$
- 4 voters: $B \succ C \succ A$
- 3 voters: $C \succ A \succ B$

a. Compute the Borda scores and identify the winner.
b. Suppose one of the 4 voters who prefers $B \succ C \succ A$ changes their ranking to $B \succ A \succ C$. What are the new Borda scores?
c. Does the outcome change? What does this tell you about the vulnerability of the Borda method to strategic voting?

Exercise 17.4:

Borda proposed the Borda Method as a response to the existence of Condorcet cycles in simply majority voting. Condorcet then presented the preference profile of Table 17.2.

Number of votes	1st choice	2nd choice	3rd choice
30	A	B	C
1	A	C	B
29	B	A	C
10	B	C	A
10	C	A	B
1	C	B	A

1. Who is the Condorcet winner (if there is one)?
2. Who is the Borda winner?
3. Why would this example be a critique of Borda's approach?

17.3 Programming

The `pref_voting` Python library (Holliday & Pacuit, 2025) provides tools for analysing preferential voting systems. In the following example, we define a preference profile based on the classroom voting scenario and use the library to explore two voting methods.

We first check for a **Condorcet winner** using pairwise majority comparisons. In this case, the method returns all alternatives, indicating that **no Condorcet winner exists** due to a cycle.

```
import pref_voting.profiles
import pref_voting.c1_methods

profile = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 18 + [[1, 2, 0]] * 16 + [[2, 0, 1]] * 11
)
print(f"Condorcet winners: {pref_voting.c1_methods.condorcet(profile)}")
```

```
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/pref_voting/
profiles.py:150: SyntaxWarning: "\l" is an invalid escape sequence. Such
```

```
sequences will not work in the future. Did you mean "\\l"? A raw string is also
an option.
  """An anonymous profile of linear rankings of :math:`n` candidates.  It is
assumed that the candidates are named :math:`0, 1, \ldots, n-1` and a ranking of
the candidates is a list of candidate names.  For instance, the list ``[0, 2,
1]`` represents the ranking in which :math:`0` is ranked
above :math:`2`, :math:`2` is ranked above :math:`1`, and :math:`0` is ranked
above :math:`1`.
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/pref_voting/
profiles.py:477: SyntaxWarning: "\l" is an invalid escape sequence. Such
sequences will not work in the future. Did you mean "\\l"? A raw string is also
an option.
  .. warning:: Since the candidates in a Profile must be named :math:`0, 1,
\ldots, n-1` (where :math:`n` is the number of candidates), you must use the
candidate map returned to by the function to recover the original candidate
names.
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/pref_voting/
profiles.py:561: SyntaxWarning: "\h" is an invalid escape sequence. Such
sequences will not work in the future. Did you mean "\\h"? A raw string is also
an option.
  latex_str += " & ".join([f"${rc}$" for rc in self._rcounts]) + "\\\\\hline \n"
```

```
Condorcet winners: [0, 1, 2]
```

Next, we apply the **Borda count**, which assigns points based on rankings and selects the alternative with the highest total score.

```
import pref_voting.scoring_methods

print(f"Borda winners: {pref_voting.scoring_methods.borda(profile)}")
```

```
/home/runner/work/gtb/gtb/.venv/lib/python3.14/site-packages/pref_voting/
scoring_methods.py:57: SyntaxWarning: "\s" is an invalid escape sequence. Such
sequences will not work in the future. Did you mean "\\s"? A raw string is also
an option.
  """The **Borda score** of a candidate is calculated as follows: If there
are :math:`m` candidates, then the Borda score of candidate :math:`c`
is :math:`\sum_{r=1}^{m} (m - r) * Rank(c,r)` where :math:`Rank(c,r)` is the
number of voters that rank candidate :math:`c` in position :math:`r`. The Borda
winners are the candidates with the largest Borda score in the ``profile``
restricted to ``curr_cands``.
```

```
Borda winners: [1]
```

17.4 Notable Research

The foundational work of Condorcet and Borda (Borda, 1781; Caritat, 1785) established key ideas in social choice theory, culminating in Arrow's impossibility theorem (Arrow, 1951). Arrow's result shows that no voting rule can simultaneously satisfy a set of seemingly reasonable fairness criteria.

These challenges deepen with the **Gibbard–Satterthwaite theorem**, which demonstrates that any non-dictatorial voting rule with at least three alternatives is vulnerable to **strategic manipulation** (Gibbard, 1973; Satterthwaite, 1975). This perhaps helps explain why some open source software

projects, such as Python, historically adopted a **Benevolent Dictator For Life** (BDFL) model of governance (O'Mahony, 2007), with Guido van Rossum serving in that role until 2018.

Encouragingly, recent work has shown that although manipulation is theoretically possible, it can be **computationally difficult in practice** (Conitzer & Sandholm, 2002; Faliszewski et al., 2010).

Other modern approaches focus on **multiwinner voting rules** that aim for fair and proportional representation (Bredereck et al., 2018; Elkind et al., 2017). These rules are increasingly relevant in applications such as committee selection, participatory budgeting, and recommendation systems.

These developments show that while Arrow's theorem rules out a perfect solution, there are many meaningful and principled ways to **navigate the trade-offs**, especially when one considers computation, uncertainty, or broader notions of fairness.

A helpful overview of these developments is provided in (Pacuit, 2019).

17.5 Conclusion

This chapter explored the central challenge of social choice: how to aggregate individual preferences into a fair and consistent group decision. Starting from a motivating classroom example, we saw how simple majority rule can lead to intransitive group preferences, known as Condorcet cycles, even when every individual's ranking is rational.

We then introduced the formal framework of preference functions and social welfare functions, leading to Arrow's impossibility theorem. This result shows that no voting rule can satisfy a basic set of fairness properties when there are at least three alternatives.

Despite this impossibility, we studied two influential responses: the **Condorcet method**, which seeks an alternative that beats all others in pairwise comparisons, and **Borda's method**, which aggregates preferences using a scoring rule. While each method has strengths and limitations, they both offer principled approaches to navigating the trade-offs inherent in group decision making.

Method	Summary	Strength	Limitation
Simple Majority	Pairwise majority voting	Intuitive and widely used	Can produce cycles
Condorcet	Chooses the option that beats all others in pairwise votes	Respects majority preferences	May not always exist
Borda	Points based on rank positions	Always yields a complete ranking	Fails to select Condorcet winner

Important

Arrow's impossibility theorem does not mean that all voting rules are equally bad. It means that trade-offs are inevitable. Understanding the strengths and limits of each method allows us to choose the right tool for the context.

17.6 Solutions

Solution 17.1: Solution to Exercise 1

The preference profile over $X = \{A, B, C\}$ is:

- 3 voters: $A \succ B \succ C$

- 2 voters: $B \succ C \succ A$
- 2 voters: $C \succ A \succ B$

There are 7 voters in total.

a. Pairwise majority contests:

A vs B:

- 3 voters prefer A to B (group 1).
- 2 voters prefer B to A (group 2).
- 2 voters prefer A to B (group 3: $C \succ A \succ B$, so $A \succ B$).

Total: A preferred by $3 + 2 = 5$ voters, B preferred by 2 voters.

A beats B with 5 votes to 2.

B vs C:

- 3 voters prefer B to C (group 1: $A \succ B \succ C$).
- 2 voters prefer B to C (group 2: $B \succ C \succ A$).
- 2 voters prefer C to B (group 3).

Total: B preferred by $3 + 2 = 5$ voters, C preferred by 2 voters.

B beats C with 5 votes to 2.

C vs A:

- 3 voters prefer A to C (group 1).
- 2 voters prefer C to A (group 2).
- 2 voters prefer C to A (group 3).

Group 1 ($A \succ B \succ C$) prefers A to C; groups 2 ($B \succ C \succ A$) and 3 ($C \succ A \succ B$) both prefer C to A. Total preferring A: 3; total preferring C: $2 + 2 = 4$.

C beats A with 4 votes to 3.

Summary of pairwise results:

- $A \underset{\text{maj}}{\succ} B$ (5 to 2)
- $B \underset{\text{maj}}{\succ} C$ (5 to 2)
- $C \underset{\text{maj}}{\succ} A$ (4 to 3)

b. Is there a Condorcet winner?

A Condorcet winner must beat all other alternatives in pairwise majority comparisons.

- A beats B but loses to C. Not a Condorcet winner.
- B beats C but loses to A. Not a Condorcet winner.
- C beats A but loses to B. Not a Condorcet winner.

There is no Condorcet winner.

c. Is the majority preference relation transitive?

The majority relation gives $A \underset{\text{maj}}{\succ} B$, $B \underset{\text{maj}}{\succ} C$, but $C \underset{\text{maj}}{\succ} A$ (not $A \underset{\text{maj}}{\succ} C$).

Transitivity would require $A \underset{\text{maj}}{\succ} C$, but we have the opposite. Therefore **the majority preference relation is not transitive**. This is a Condorcet cycle: A beats B, B beats C, C beats A.

```
import pref_voting.profiles
import pref_voting.c1_methods

# Encode: A=0, B=1, C=2
profile = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 3 + [[1, 2, 0]] * 2 + [[2, 0, 1]] * 2
)
condorcet_winner = pref_voting.c1_methods.condorcet(profile)
print("Condorcet winner(s):", condorcet_winner)
```

```
Condorcet winner(s): [0, 1, 2]
```

```
# Pairwise margin matrix
import numpy as np

# voters[i] = list of preferences in order (most to least preferred)
votes = [[0, 1, 2]] * 3 + [[1, 2, 0]] * 2 + [[2, 0, 1]] * 2
n_alt = 3
margin = np.zeros((n_alt, n_alt), dtype=int)
for v in votes:
    for i in range(n_alt):
        for j in range(i + 1, n_alt):
            x, y = v[i], v[j]
            margin[x][y] += 1
            margin[y][x] -= 1

labels = ["A", "B", "C"]
print("Pairwise margins (row beats column by this many votes):")
for i in range(n_alt):
    for j in range(n_alt):
        if i != j:
            print(f"  {labels[i]} vs {labels[j]}: {margin[i][j]:+d}")
```

```
Pairwise margins (row beats column by this many votes):
  A vs B: +3
  A vs C: -1
  B vs A: -3
  B vs C: +3
  C vs A: +1
  C vs B: -3
```

Solution 17.2: Solution to Exercise 2

The preference profile is:

- 4 voters: $A \succ B \succ C$
- 3 voters: $B \succ C \succ A$
- 2 voters: $C \succ A \succ B$

There are 9 voters in total.

a. Borda scores:

With three alternatives, each voter assigns 2 points to their first choice, 1 point to their second choice, and 0 points to their last choice.

- **A:** $4 \times 2 + 3 \times 0 + 2 \times 1 = 8 + 0 + 2 = 10$
- **B:** $4 \times 1 + 3 \times 2 + 2 \times 0 = 4 + 6 + 0 = 10$
- **C:** $4 \times 0 + 3 \times 1 + 2 \times 2 = 0 + 3 + 4 = 7$

Borda scores: $A = 10$, $B = 10$, $C = 7$.

The Borda method produces a **tie between A and B**. (Without a tie-breaking rule, neither is uniquely selected.)

b. Condorcet winner:

A vs B:

- 4 voters prefer A (group 1).
- 3 voters prefer B (group 2).
- 2 voters prefer A (group 3: $C \succ A \succ B$, so $A \succ B$).

A preferred by $4 + 2 = 6$, B preferred by 3. **A beats B 6 to 3.**

A vs C:

- 4 voters prefer A (group 1).
- 3 voters prefer C (group 2).
- 2 voters prefer C (group 3).

A preferred by 4, C preferred by $3 + 2 = 5$. **C beats A 5 to 4.**

B vs C:

- 4 voters prefer B (group 1).
- 3 voters prefer B (group 2).
- 2 voters prefer C (group 3).

B preferred by $4 + 3 = 7$, C preferred by 2. **B beats C 7 to 2.**

Pairwise results: A beats B; B beats C; C beats A. This is again a Condorcet cycle. **There is no Condorcet winner.**

c. Do Borda and Condorcet select the same winner?

There is no Condorcet winner here (due to the cycle). The Borda method yields a tie between A and B. Since no Condorcet winner exists, we cannot directly compare the two methods in terms of who they "select", but we can note that the Borda method at least produces a complete ranking and is decisive (up to ties), while the Condorcet method fails to identify a winner in this case.

```
import pref_voting.profiles
import pref_voting.c1_methods
import pref_voting.scoring_methods

profile = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 4 + [[1, 2, 0]] * 3 + [[2, 0, 1]] * 2
```

```
)
print("Condorcet winner:", pref_voting.c1_methods.condorcet(profile))
print("Borda winner:", pref_voting.scoring_methods.borda(profile))
```

```
Condorcet winner: [0, 1, 2]
Borda winner: [0, 1]
```

```
# Manual Borda score computation
groups = [(4, [0, 1, 2]), (3, [1, 2, 0]), (2, [2, 0, 1])]
n_alt = 3
borda_scores = [0, 0, 0]
for count, ranking in groups:
    for rank, alt in enumerate(ranking):
        borda_scores[alt] += count * (n_alt - 1 - rank)
labels = ["A", "B", "C"]
for label, score in zip(labels, borda_scores):
    print(f"Borda score of {label}: {score}")
```

```
Borda score of A: 10
Borda score of B: 10
Borda score of C: 7
```

Solution 17.3: Solution to Exercise 3

The preference profile is:

- 5 voters: $A \succ B \succ C$
- 4 voters: $B \succ C \succ A$
- 3 voters: $C \succ A \succ B$

There are 12 voters in total.

a. Original Borda scores:

With three alternatives each voter assigns 2 points to their top choice, 1 to second, 0 to last.

- **A:** $5 \times 2 + 4 \times 0 + 3 \times 1 = 10 + 0 + 3 = 13$
- **B:** $5 \times 1 + 4 \times 2 + 3 \times 0 = 5 + 8 + 0 = 13$
- **C:** $5 \times 0 + 4 \times 1 + 3 \times 2 = 0 + 4 + 6 = 10$

Borda scores: $A = 13$, $B = 13$, $C = 10$.

The **Borda winner is a tie between A and B**.

b. Manipulation: one voter changes from $B \succ C \succ A$ to $B \succ A \succ C$:

The new profile is:

- 5 voters: $A \succ B \succ C$
- 3 voters: $B \succ C \succ A$ (one fewer)
- 1 voter: $B \succ A \succ C$ (the manipulating voter)
- 3 voters: $C \succ A \succ B$

New Borda scores:

- **A:** $5 \times 2 + 3 \times 0 + 1 \times 1 + 3 \times 1 = 10 + 0 + 1 + 3 = 14$
- **B:** $5 \times 1 + 3 \times 2 + 1 \times 2 + 3 \times 0 = 5 + 6 + 2 + 0 = 13$
- **C:** $5 \times 0 + 3 \times 1 + 1 \times 0 + 3 \times 2 = 0 + 3 + 0 + 6 = 9$

New Borda scores: $A = 14$, $B = 13$, $C = 9$.

c. Does the outcome change?

Originally A and B were tied (both at 13). After the manipulation, $\boldsymbol{A}$ **wins outright with 14 points**, while B drops to 13.

The manipulating voter prefers B to A, yet their manipulation caused A to win (or more precisely, broke the tie in A's favour, away from B). This shows that the manipulation was **not profitable for this voter**; it was inadvertent harm.

However, the fact that changing a ballot in a way that does not affect the voter's stated preference between B and C (they still rank B first) can nonetheless change the winner illustrates the Borda method's **vulnerability to strategic voting**. A voter who, in a different scenario, wanted to harm A could promote C (an irrelevant alternative) to second place, boosting C's score and potentially tipping the outcome. This violates the **Independence of Irrelevant Alternatives** property and is the mechanism through which manipulation operates.

More broadly, the Borda method is susceptible to strategic manipulation because the relative ranking of any two alternatives can be influenced by how a voter ranks a third, "irrelevant" alternative. This is precisely what the Gibbard-Satterthwaite theorem predicts: any non-dictatorial voting rule over three or more alternatives is manipulable.

```
import pref_voting.profiles
import pref_voting.scoring_methods

# Original profile: A=0, B=1, C=2
profile_original = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 5 + [[1, 2, 0]] * 4 + [[2, 0, 1]] * 3
)
print("Original Borda winner:",
pref_voting.scoring_methods.borda(profile_original))
```

```
Original Borda winner: [0, 1]
```

```
# Manipulated profile: one voter switches from B>C>A to B>A>C
profile_manipulated = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 5 + [[1, 2, 0]] * 3 + [[1, 0, 2]] * 1 + [[2, 0, 1]] * 3
)
print("Manipulated Borda winner:",
pref_voting.scoring_methods.borda(profile_manipulated))
```

```
Manipulated Borda winner: [0]
```

```
# Manual score verification
def borda_scores(groups, n_alt=3):
    scores = [0] * n_alt
    for count, ranking in groups:
        for rank, alt in enumerate(ranking):
```

```
            scores[alt] += count * (n_alt - 1 - rank)
    return scores

original = borda_scores([(5,[0,1,2]),(4,[1,2,0]),(3,[2,0,1])])
manipulated = borda_scores([(5,[0,1,2]),(3,[1,2,0]),(1,[1,0,2]),(3,[2,0,1])])
labels = ["A","B","C"]
print("Original scores:", {l:s for l,s in zip(labels, original)})
print("Manipulated scores:", {l:s for l,s in zip(labels, manipulated)})
```

```
Original scores: {'A': 13, 'B': 13, 'C': 10}
Manipulated scores: {'A': 14, 'B': 13, 'C': 9}
```

Solution 17.4: Solution to Exercise 4

The preference profile is:

Number of votes	1st	2nd	3rd
30	A	B	C
1	A	C	B
29	B	A	C
10	B	C	A
10	C	A	B
1	C	B	A

Total voters: $30 + 1 + 29 + 10 + 10 + 1 = 81$.

1. Who is the Condorcet winner?

We check each pairwise majority contest.

A vs B:

Voters preferring A over B: groups ranking A above B:

- 30 voters ($A \succ B \succ C$)
- 1 voter ($A \succ C \succ B$)
- 10 voters ($C \succ A \succ B$)

Total preferring A: $30 + 1 + 10 = 41$.

Voters preferring B over A:

- 29 voters ($B \succ A \succ C$)
- 10 voters ($B \succ C \succ A$)
- 1 voter ($C \succ B \succ A$)

Total preferring B: $29 + 10 + 1 = 40$.

***A* beats *B* with 41 to 40.**

A vs C:

Voters preferring A over C:

- 30 voters ($A \succ B \succ C$)
- 1 voter ($A \succ C \succ B$)
- 29 voters ($B \succ A \succ C$)

Total: $30 + 1 + 29 = 60$.

Voters preferring C over A:

- 10 voters ($B \succ C \succ A$)
- 10 voters ($C \succ A \succ B$)
- 1 voter ($C \succ B \succ A$)

Total: $10 + 10 + 1 = 21$.

A beats C with 60 to 21.

B vs C:

Voters preferring B over C:

- 30 voters ($A \succ B \succ C$)
- 29 voters ($B \succ A \succ C$)
- 10 voters ($B \succ C \succ A$)

Voters preferring C over B:

- 1 voter ($A \succ C \succ B$)
- 10 voters ($C \succ A \succ B$)
- 1 voter ($C \succ B \succ A$)

Total preferring B: $30 + 29 + 10 = 69$. Total preferring C: $1 + 10 + 1 = 12$.

B beats C with 69 to 12.

A beats both B and C, so **A is the Condorcet winner**.

2. Who is the Borda winner?

With three alternatives, each voter assigns 2 points to their 1st choice, 1 point to their 2nd choice, 0 to their 3rd choice.

Group (count)	A points	B points	C points
30 ($A \succ B \succ C$)	$30 \times 2 = 60$	$30 \times 1 = 30$	$30 \times 0 = 0$
1 ($A \succ C \succ B$)	$1 \times 2 = 2$	$1 \times 0 = 0$	$1 \times 1 = 1$
29 ($B \succ A \succ C$)	$29 \times 1 = 29$	$29 \times 2 = 58$	$29 \times 0 = 0$
10 ($B \succ C \succ A$)	$10 \times 0 = 0$	$10 \times 2 = 20$	$10 \times 1 = 10$
10 ($C \succ A \succ B$)	$10 \times 1 = 10$	$10 \times 0 = 0$	$10 \times 2 = 20$
1 ($C \succ B \succ A$)	$1 \times 0 = 0$	$1 \times 1 = 1$	$1 \times 2 = 2$

Total Borda scores:

- **A:** $60 + 2 + 29 + 0 + 10 + 0 = 101$
- **B:** $30 + 0 + 58 + 20 + 0 + 1 = 109$
- **C:** $0 + 1 + 0 + 10 + 20 + 2 = 33$

The **Borda winner is B** with 109 points.

3. Why is this a critique of Borda's method?

A is the Condorcet winner: it beats every other alternative in pairwise majority votes (41 to 40 over B; 60 to 21 over C). A majority of voters prefers A to B (41 out of 81).

Yet the Borda method selects B as the winner.

This is Condorcet's critique of the Borda method: **the Borda method can fail to elect the Condorcet winner** even when one exists. The method gives weight to the intensity of preferences (how far apart alternatives are ranked) rather than just pairwise majority relationships. Here, B benefits from being ranked second by many voters (the 30 who rank A first and the 10 who rank C first), accumulating enough second-place votes to overtake A's lead in direct comparisons.

This example showed Condorcet that the Borda method violates the Condorcet criterion, a property that many would argue is a minimal requirement of a reasonable voting rule.

```
import pref_voting.profiles
import pref_voting.c1_methods
import pref_voting.scoring_methods

# A=0, B=1, C=2
profile = pref_voting.profiles.Profile(
    [[0, 1, 2]] * 30
    + [[0, 2, 1]] * 1
    + [[1, 0, 2]] * 29
    + [[1, 2, 0]] * 10
    + [[2, 0, 1]] * 10
    + [[2, 1, 0]] * 1
)

print("Condorcet winner:", pref_voting.c1_methods.condorcet(profile))
print("Borda winner:", pref_voting.scoring_methods.borda(profile))
```

```
Condorcet winner: [0]
Borda winner: [1]
```

```
# Manual Borda score verification
groups = [
    (30, [0, 1, 2]),
    (1,  [0, 2, 1]),
    (29, [1, 0, 2]),
    (10, [1, 2, 0]),
    (10, [2, 0, 1]),
    (1,  [2, 1, 0]),
]
scores = [0, 0, 0]
for count, ranking in groups:
    for rank, alt in enumerate(ranking):
        scores[alt] += count * (2 - rank)
print("Borda scores. A:", scores[0], "B:", scores[1], "C:", scores[2])
```

```
Borda scores. A: 101 B: 109 C: 33
```

18. Cooperative Games

When players can form coalitions and make binding agreements, the key question shifts from what strategy to play to how to share the resulting surplus fairly. This chapter studies cooperative games through the characteristic function and develops the Shapley value as a principled allocation scheme.

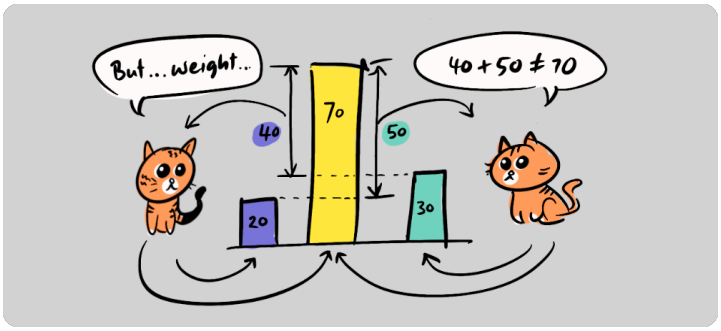


Figure 18.1: How much value does each player add to a cooperative effort? The marginal contribution measures exactly this, and averaging it over every order of joining yields the Shapley value, a principled way to share the surplus.

18.1 Motivating Example: Dividing the loot after a D&D battle

After a climactic boss fight, a party of adventurers must divide a hoard of 1,000 gold coins recovered after fighting a dragon.

The party includes:

- **Ren** the fighter, who absorbed the dragon's attacks.
- **Azel** the wizard, who cast a decisive spell at the turning point.
- **Quinn** the bard, who provided support and distracted the dragon

with an improvised limerick.

Each character played a crucial role, but in very different ways. How can they divide the treasure fairly?

Based on past encounters, the party has a sense of how much gold they might have secured without the full team:

- $\{\text{Ren}\} = 0$ (they can't get past the dragon alone)
- $\{\text{Azel}\} = 0$ (powerful spells, but too vulnerable on their own)
- $\{\text{Quinn}\} = 0$ (charming, but not a serious threat)
- $\{\text{Ren}, \text{Azel}\} = 900$ (they can defeat the dragon, but with great effort)
- $\{\text{Ren}, \text{Quinn}\} = 400$ (they can stall the dragon,

but not defeat it)

- $\{\text{Azel}, \text{Quinn}\} = 600$ (Azel can burn the dragon while

Quinn keeps it distracted)

- $\{\text{Ren}, \text{Azel}, \text{Quinn}\} = 1000$ (the full party defeats

the dragon smoothly and decisively)

This is a classic example of a **cooperative game**, and understanding how to allocate resources fairly in such settings is the focus of this chapter.

18.2 Theory

18.2.1 Definition: characteristic function game

A **characteristic function game** G is given by a pair (N, v) where N is the number of players and $v : 2^{[N]} \to \mathbb{R}$ is a **characteristic function** which maps every coalition of players to a payoff.

18.2.1.1 Example: The characteristic function game of the D&D battle

For the D&D battle, we have $N = 3$ and the characteristic function v is given by:

$$v(c) = \tag{18.1}$$

18.2.2 Definition: Monotone characteristic function game

A characteristic function game $G = (N, v)$ is called **monotone** if it satisfies $v(C_2) \geq v(C_1)$ for all $C_1 \subseteq C_2$.

18.2.2.1 Example: The D&D battle is a monotone characteristic function game

The function defined in (18.1) is monotone.
However, the following game is not:

$$v(c) = \tag{18.2}$$

Indeed, Ren and Azel receive more **without** Quinn than with them, violating monotonicity.

18.2.3 Definition: superadditive characteristic function game

A characteristic function game $G = (N, v)$ is called **superadditive** if it satisfies $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$ for all disjoint coalitions $C_1 \cap C_2 = \varnothing$.

18.2.3.1 Example: The D&D battle is not a superadditive characteristic function game

The function defined in (18.1) is superadditive.
However, the following game is not:

$$v(c) = \tag{18.3}$$

Indeed:

$$v(\{\text{Ren}, \text{Quinn}\}) + v(\{\text{Azel}\}) = 400 + 700 = 1100 > 1000 = v(\{\text{Ren}, \text{Azel}, \text{Quinn}\}) \tag{18.4}$$

so the game violates superadditivity.

18.2.4 Definition: Payoff vector

When we talk about a solution to a characteristic function game, we mean a payoff vector $\lambda \in \mathbb{R}^N_{\geq 0}$ that allocates the value of the grand coalition among the players. In particular, λ must satisfy:

$$\sum_{i=1}^{N} \lambda_i = v(\Omega) \tag{18.5}$$

18.2.4.1 Example: A possible payoff vector:

One possible payoff vector for (18.1) is:

$$\lambda = (200, 350, 450) \tag{18.6}$$

This would seem unfair as Quinn gets 450 of the gold. To find a "fair" distribution of the grand coalition we must define what is meant by "fair". We require four desirable properties:

- Efficiency;
- Null player;
- Symmetry;
- Additivity.

18.2.5 Definition: Efficiency

Given a game $G = (N, v)$, a payoff vector λ is **efficient** if:

$$\sum_{i=1}^{N} \lambda_i = v(\Omega) \tag{18.7}$$

18.2.5.1 Example: An efficient Payoff Vector for the D&D Battle

An efficient payoff vector is one that shares all 1,000 gold coins like the one in (18.6).

18.2.6 Definition: Null Player Property

Given a game $G = (N, v)$, a payoff vector λ satisfies the **null player property** if, for a player i:

If $v(C \cup i) = v(C)$ for all coalitions $C \subseteq \Omega$, then:

$$\lambda_i = 0 \tag{18.8}$$

18.2.6.1 Example: The Null Player Property in the D&D Battle

In Motivating Example: Dividing the loot after a D&D battle, Quinn contributes very little, yet still adds value to each coalition he joins. For instance, without Quinn, the grand coalition would only obtain 900 coins. If, however, a member of the party made no difference to the value of any coalition, the null player property dictates that this player should receive a payoff of 0.

18.2.7 Definition: Symmetry Property

Given a game $G = (N, v)$, a payoff vector λ satisfies the **symmetry property** if, for any pair of players i, j:

If $v(C \cup i) = v(C \cup j)$ for all coalitions $C \subseteq \Omega \setminus \{i, j\}$, then:

$$\lambda_i = \lambda_j \tag{18.9}$$

18.2.7.1 Example: The Symmetry Property in the D&D Battle

In Motivating Example: Dividing the loot after a D&D battle, no two players contribute equally to every coalition they join. If there were such players, the symmetry property would guarantee that they each receive an equal share of the gold

18.2.8 Definition: Additivity Property

Given two games $G_1 = (N, v_1)$ and $G_2 = (N, v_2)$, define their sum $G^+ = (N, v^+)$ by:

$$v^+(C) = v_1(C) + v_2(C) \quad \text{for all } C \subseteq \Omega \tag{18.10}$$

A payoff vector λ satisfies the **additivity property** if:

$$\lambda_i^{(G^+)} = \lambda_i^{(G_1)} + \lambda_i^{(G_2)} \tag{18.11}$$

18.2.8.1 Example: The Additive Property in the D&D Battle

In Motivating Example: Dividing the loot after a D&D battle, if there were in fact two separate battles, the method for distributing the gold should
yield the same result as treating both battles as a single event with the
combined total reward.

18.2.9 Definition: Predecessors

Given a permutation π of $[N]$, the set of **predecessors** of player i in π is denoted by $S_p i(i)$ and defined as:

$$S_\pi(i) = \{j \in [N] \texttt{ mid } \pi(j) < \pi(i)\} \tag{18.12}$$

18.2.9.1 Example: The predecessors for the D&D Battle

For Motivating Example: Dividing the loot after a D&D battle we have $N = 3$ if we assume the ordering: (Ren, Azel, Quinn) Table 18.1 gives the predecessors of each individual for each of the $3! = 6$ permutations of $[N]$.

π	$S_\pi(1)$ (Predecessors of Ren)	$S_\pi(2)$ (Predecessors of Azel)	$S_\pi(3)$ (Predecessors of Quinn)
$(1, 2, 3)$	$\varnothing$	$\{1\}$	$\{1, 2\}$
$(1, 3, 2)$	$\varnothing$	$\{1, 3\}$	$\{1\}$
$(2, 1, 3)$	$\{2\}$	$\varnothing$	$\{1, 2\}$
$(2, 3, 1)$	$\{2, 3\}$	$\varnothing$	$\{2\}$
$(3, 1, 2)$	$\{3\}$	$\{1, 3\}$	$\varnothing$
$(3, 2, 1)$	$\{2, 3\}$	$\{3\}$	$\varnothing$

Table 18.1: The predecessors of each individual for each permutation

Note

Here we are using the indices of the players for ease of notation where 1 corresponds to the first individual (Ren), 2 corresponds to the second individual (Azel) and 3 corresponds to the third individual $Quinn$.

18.2.10 Definition: Marginal Contribution

Given a permutation π of $[N]$, the **marginal contribution** of player i with respect to π is defined as:

$$\Delta_\pi^G(i) = v(S_\pi(i) \cup \{i\}) - v(S_\pi(i)) \tag{18.13}$$

18.2.10.1 Example: Marginal Contributions in the D&D Battle

For Motivating Example: Dividing the loot after a D&D battle (Ren, Azel, Quinn), with $N = 3$. Table 18.2 shows the marginal contribution $\Delta_\pi^G(i)$ for each player i in each permutation π using (18.1).

π	$\Delta_\pi^G(1)$ (Ren)	$\Delta_\pi^G(2)$ (Azel)	$\Delta_\pi^G(3)$ (Quinn)
$(1,2,3)$	0	900	100
$(1,3,2)$	0	600	400
$(2,1,3)$	900	0	100
$(2,3,1)$	400	0	600
$(3,1,2)$	400	600	0
$(3,2,1)$	400	600	0

Table 18.2: Marginal contributions of each individual for each permutation

We are now ready to define the **Shapley value** for any cooperative game $G = (N, v)$.

18.2.11 Definition: Shapley Value

Given a cooperative game $G = (N, v)$, the **Shapley value** of player i is denoted by $\phi_i(G)$ and defined as:

$$\phi_i(G) = \frac{1}{N!} \sum_{\pi \in \Pi_n} \Delta_\pi^G(i) \tag{18.14}$$

18.2.11.1 Example: The Shapley Value for the D&D Battle

For Motivating Example: Dividing the loot after a D&D battle the Shapley value is the vector with entries corresponding to the average of each of the columnts of Table 18.2.

$$\begin{aligned} \phi_1 &= \frac{0+0+900+400+400+400}{6} = 350 \\ \phi_2 &= \frac{900+600+0+0+600+600}{6} = 450 \\ \phi_3 &= \frac{100+400+100+600+0+0}{6} = 200 \end{aligned} \tag{18.15}$$

Giving: $\phi = (350, 450, 200)$

18.3 Exercises

Exercise 18.1:

For the following cooperative games:

1. Verify whether the game is monotonic.
2. Verify whether the game is superadditive.
3. Obtain the Shapley value.

$$v_1(C) = \tag{18.16}$$

$$v_2(C) = \tag{18.17}$$

$$v_3(C) = \tag{18.18}$$

$$v_4(C) = \tag{18.19}$$

Exercise 18.2:

In statistics and machine learning, a **linear model** predicts a target variable y as a weighted sum of input features $x_1, x_2, ..., x_n$. The **coefficient of determination**, denoted R^2, measures how well the model explains the variance in the data. It takes values between 0 and 1, with higher values indicating better explanatory power.

In cooperative game theory, a **characteristic function** maps every coalition (subset of players) to a value. In this exercise, we will interpret each feature as a player, and define the value of a coalition to be the R^2 of the model trained on those features.

Suppose we are predicting a target variable y using a linear model:

$$y = c_1x_1 + c_2x_2 + c_3x_3 \tag{18.20}$$

Below are the R^2 values obtained from fitting models to different subsets of the features. These were generated using synthetic data.

Model	R^2
$y = c_1x_1$	0.122
$y = c_2x_2$	0.097
$y = c_3x_3$	0.551
$y = c_1x_1 + c_2x_2$	0.174
$y = c_1x_1 + c_3x_3$	0.581
$y = c_2x_2 + c_3x_3$	0.620
$y = c_1x_1 + c_2x_2 + c_3x_3$	0.623

1. Define the characteristic function $v(S)$ for each subset of $\{x_1, x_2, x_3\}$ using the table above.
2. Compute the Shapley value for each feature with respect to the characteristic function v.
3. Interpret the result: Which feature contributes the most explanatory power? Which contributes the least?

Exercise 18.3:

Consider the following two cooperative games defined on $N = \{1, 2, 3\}$:

Game v_a:

$$v_a(C) = \tag{18.21}$$

Game v_b:

$$v_b(C) = \tag{18.22}$$

1. Verify that both games satisfy the symmetry property.
2. Obtain the Shapley value for v_a and v_b individually.

3. Construct the game $(v_a + v_b)$ and calculate the Shapley value.
4. Verify that the Shapley value of $(v_a + v_b)$ equals the sum of the Shapley values of v_a and v_b.

Exercise 18.4:

Consider the cooperative game v defined on $N = \{1, 2, 3\}$:

$$v(C) = \tag{18.23}$$

1. Identify any null players in this game.
2. Compute the marginal contributions of each player.
3. Calculate the Shapley value and confirm it respects the null player property.

Exercise 18.5:

Prove that the Shapley value satisfies the following properties:

- Efficiency
- Null player property
- Symmetry
- Additivity

Note that this does not prove uniqueness; that is, other vectors might satisfy these properties (though in fact, the Shapley value is uniquely defined by them).

18.4 Programming

The CoopGT library has functionality to verify properties of a characteristic function game and also compute the Shapley value.

Here let us define a characteristic function and check if it is valid:

```
import coopgt.characteristic_function_properties


characteristic_function = {
    (): 0,
    (1,): 0,
    (2,): 0,
    (3,): 0,
    (1, 2): 900,
    (1,3): 400,
    (2, 3): 600,
    (1, 2, 3): 1000,
}
is_valid = coopgt.characteristic_function_properties.is_valid(
    characteristic_function=characteristic_function
)
print(f"Characteristic function valid? {is_valid}")
```

```
Characteristic function valid? True
```

Here we can check if it is monotone:

```
is_monotone = coopgt.characteristic_function_properties.is_monotone(
    characteristic_function=characteristic_function
)
print(f"Characteristic function monotone? {is_monotone}")
```

```
Characteristic function monotone? True
```

Here we can check if it is supperadditive:

```
is_superadditive = coopgt.characteristic_function_properties.is_superadditive(
    characteristic_function=characteristic_function
)
print(f"Characteristic function superadditive? {is_superadditive}")
```

```
Characteristic function superadditive? True
```

Let us compute the Shapley value:

```
import coopgt.shapley_value

shapley_value = coopgt.shapley_value.calculate(
    characteristic_function=characteristic_function
)
print(f"Shapley value: {shapley_value}")
```

```
Shapley value: [350. 450. 200.]
```

18.5 Notable Research

The Shapley value was first introduced by Shapley in (Shapley & others, 1953), a foundational paper in cooperative game theory. Although Shapley is best known for this concept, he was awarded the Nobel Prize in Economics for his later work on matching games.

In large games, the computational cost of calculating the Shapley value can be prohibitive. While not necessarily intractable in the formal sense, as shown in (Deng & Papadimitriou, 1994), the number of permutations makes exact computation impractical. One of the earliest practical approaches to this challenge is (Castro et al., 2009), which proposes Monte Carlo sampling of permutations as an efficient approximation method.

One of the most impactful application areas of the Shapley value in recent years has been in **model explainability**. As explored in Exercise 2, the Shapley value provides a principled way to attribute a model's output to its input features. One of the first papers to formalise this idea was (Strumbelj & Kononenko, 2010). This approach has since seen wide application, particularly in healthcare, where a comprehensive overview is provided in (Jin et al., 2022).

18.6 Conclusion

In this chapter, we introduced the theory of **cooperative games**, where groups of players can form coalitions and share the resulting value. We focused on **characteristic function games**, where the value of each coalition is known, and explored how to fairly distribute this value using the **Shapley value**.

The four axioms (efficiency, null player, symmetry, and additivity) uniquely determine the Shapley value. We also saw applications to machine learning interpretability.

Table 18.3 gives a summary of the concepts of this chapter.

Concept	Description
Characteristic function	Maps each subset (coalition) of players to a real-valued payoff
Monotonicity	Adding players to a coalition cannot decrease its value
Superadditivity	The value of two disjoint coalitions is at most the value of their union
Payoff vector	A distribution of the total value among all players
Marginal contribution	The added value a player brings to a coalition
Predecessors in permutation	Players who appear before a given player in a permutation
Shapley value	The average marginal contribution: a unique, fair payoff satisfying four axioms

Table 18.3: Summary of cooperative game concepts

Important

The Shapley value is the only payoff rule that satisfies **efficiency**, **null player**, **symmetry**, and **additivity**, making it a principled way to fairly divide rewards in cooperative settings.

18.7 Solutions

Solution 18.1: Solution to Exercise 1

Game v_1

$$v_1(C) = \tag{18.24}$$

1. **Monotonicity of v_1**: We must check $v(C_1) \leq v(C_2)$ for all $C_1 \subseteq C_2$. All singleton-to-pair comparisons:
   - $v_1(\{1\}) = 5 \leq 12 = v_1(\{1,2\})$ ✓
   - $v_1(\{1\}) = 5 \leq 5 = v_1(\{1,3\})$ ✓
   - $v_1(\{2\}) = 3 \leq 12 = v_1(\{1,2\})$ ✓
   - $v_1(\{2\}) = 3 \leq 4 = v_1(\{2,3\})$ ✓
   - $v_1(\{3\}) = 2 \leq 5 = v_1(\{1,3\})$ ✓
   - $v_1(\{3\}) = 2 \leq 4 = v_1(\{2,3\})$ ✓

All pair-to-grand-coalition comparisons:

- $v_1(\{1,2\}) = 12 \leq 13 = v_1(\{1,2,3\})$ ✓
- $v_1(\{1,3\}) = 5 \leq 13 = v_1(\{1,2,3\})$ ✓
- $v_1(\{2,3\}) = 4 \leq 13 = v_1(\{1,2,3\})$ ✓

v_1 is monotone.

1. **Superadditivity of v_1**: We check $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$ for all disjoint C_1, C_2.
   - $v_1(\{1,2\}) \geq v_1(\{1\}) + v_1(\{2\})$: $12 \geq 5 + 3 = 8$ ✓
   - $v_1(\{1,3\}) \geq v_1(\{1\}) + v_1(\{3\})$: $5 \geq 5 + 2 = 7$? No: $5 < 7$.

v_1 is not superadditive (coalition $\{1,3\}$ is worth less than the sum of the individual values of players 1 and 3).

1. **Shapley value of v_1**: We enumerate all $3! = 6$ permutations and compute marginal contributions.

π	$\Delta_\pi^{v_1}(1)$	$\Delta_\pi^{v_1}(2)$	$\Delta_\pi^{v_1}(3)$
$(1,2,3)$	$v_1(\{1\}) - v_1(\varnothing) = 5$	$v_1(\{1,2\}) - v_1(\{1\}) = 7$	$v_1(\{1,2,3\}) - v_1(\{1,2\}) = 1$
$(1,3,2)$	5	$v_1(\{1,2,3\}) - v_1(\{1,3\}) = 8$	$v_1(\{1,3\}) - v_1(\{1\}) = 0$
$(2,1,3)$	$v_1(\{1,2\}) - v_1(\{2\}) = 9$	3	$v_1(\{1,2,3\}) - v_1(\{1,2\}) = 1$
$(2,3,1)$	$v_1(\{1,2,3\}) - v_1(\{2,3\}) = 9$	3	$v_1(\{2,3\}) - v_1(\{2\}) = 1$
$(3,1,2)$	$v_1(\{1,3\}) - v_1(\{3\}) = 3$	$v_1(\{1,2,3\}) - v_1(\{1,3\}) = 8$	2
$(3,2,1)$	$v_1(\{1,2,3\}) - v_1(\{2,3\}) = 9$	$v_1(\{2,3\}) - v_1(\{3\}) = 2$	2

$$\phi_1(v_1) = \frac{5+5+9+9+3+9}{6} = \frac{40}{6} = \frac{20}{3} \approx 6.67 \tag{18.25}$$

$$\phi_2(v_1) = \frac{7+8+3+3+8+2}{6} = \frac{31}{6} \approx 5.17 \tag{18.26}$$

$$\phi_3(v_1) = \frac{1+0+1+1+2+2}{6} = \frac{7}{6} \approx 1.17 \tag{18.27}$$

Check: $\phi_1 + \phi_2 + \phi_3 = 20/3 + 31/6 + 7/6 = 40/6 + 31/6 + 7/6 = 78/6 = 13 = v_1(\{1,2,3\})$ ✓

Game v_2

$$v_2(C) = \tag{18.28}$$

This is a two-player game with $N = 2$.

1. **Monotonicity of v_2**:
   - $v_2(\{1\}) = 6 \leq 5 = v_2(\{1,2\})$? No: $6 > 5$.

v_2 is not monotone.

1. **Superadditivity of v_2**:
   - $v_2(\{1,2\}) \geq v_2(\{1\}) + v_2(\{2\})$: $5 \geq 6 + 0 = 6$? No: $5 < 6$.

v_2 is not superadditive.

1. **Shapley value of v_2**: With $N = 2$ there are $2! = 2$ permutations.

π	$\Delta_\pi^{v_2}(1)$	$\Delta_\pi^{v_2}(2)$
$(1,2)$	$v_2(\{1\}) - v_2(\varnothing) = 6$	$v_2(\{1,2\}) - v_2(\{1\}) = -1$
$(2,1)$	$v_2(\{1,2\}) - v_2(\{2\}) = 5$	$v_2(\{2\}) - v_2(\varnothing) = 0$

$$\phi_1(v_2) = \frac{6+5}{2} = \frac{11}{2} = 5.5 \tag{18.29}$$

$$\phi_2(v_2) = \frac{-1+0}{2} = -\frac{1}{2} = -0.5 \tag{18.30}$$

Check: $5.5 + (-0.5) = 5 = v_2(\{1,2\})$ ✓

Game v_3

$$v_3(C) = \tag{18.31}$$

1. **Monotonicity of v_3**:
   - $v_3(\{1\}) = 6 \leq 6 = v_3(\{1,2\})$ ✓
   - $v_3(\{2\}) = 6 \leq 6 = v_3(\{1,2\})$ ✓
   - $v_3(\{1\}) = 6 \leq 13 = v_3(\{1,3\})$ ✓
   - $v_3(\{3\}) = 13 \leq 13 = v_3(\{1,3\})$ ✓
   - $v_3(\{2\}) = 6 \leq 13 = v_3(\{2,3\})$ ✓
   - $v_3(\{3\}) = 13 \leq 13 = v_3(\{2,3\})$ ✓
   - $v_3(\{1,2\}) = 6 \leq 26 = v_3(\{1,2,3\})$ ✓
   - $v_3(\{1,3\}) = 13 \leq 26 = v_3(\{1,2,3\})$ ✓
   - $v_3(\{2,3\}) = 13 \leq 26 = v_3(\{1,2,3\})$ ✓

v_3 is monotone.

1. **Superadditivity of v_3**:
   - $v_3(\{1,2\}) \geq v_3(\{1\}) + v_3(\{2\})$: $6 \geq 6 + 6 = 12$? No: $6 < 12$.

v_3 is not superadditive.

1. **Shapley value of v_3**:

π	$\Delta_\pi^{v_3}(1)$	$\Delta_\pi^{v_3}(2)$	$\Delta_\pi^{v_3}(3)$
$(1,2,3)$	6	$v_3(\{1,2\}) - v_3(\{1\}) = 0$	$v_3(\{1,2,3\}) - v_3(\{1,2\}) = 20$
$(1,3,2)$	6	$v_3(\{1,2,3\}) - v_3(\{1,3\}) = 13$	$v_3(\{1,3\}) - v_3(\{1\}) = 7$
$(2,1,3)$	$v_3(\{1,2\}) - v_3(\{2\}) = 0$	6	$v_3(\{1,2,3\}) - v_3(\{1,2\}) = 20$
$(2,3,1)$	$v_3(\{1,2,3\}) - v_3(\{2,3\}) = 13$	6	$v_3(\{2,3\}) - v_3(\{2\}) = 7$
$(3,1,2)$	$v_3(\{1,3\}) - v_3(\{3\}) = 0$	$v_3(\{1,2,3\}) - v_3(\{1,3\}) = 13$	13
$(3,2,1)$	$v_3(\{1,2,3\}) - v_3(\{2,3\}) = 13$	$v_3(\{2,3\}) - v_3(\{3\}) = 0$	13

$$\phi_1(v_3) = \frac{6+6+0+13+0+13}{6} = \frac{38}{6} = \frac{19}{3} \approx 6.33 \tag{18.32}$$

$$\phi_2(v_3) = \frac{0+13+6+6+13+0}{6} = \frac{38}{6} = \frac{19}{3} \approx 6.33 \tag{18.33}$$

$$\phi_3(v_3) = \frac{20+7+20+7+13+13}{6} = \frac{80}{6} = \frac{40}{3} \approx 13.33 \tag{18.34}$$

Check: $19/3 + 19/3 + 40/3 = 78/3 = 26 = v_3(\{1,2,3\})$ ✓

Note that $\phi_1 = \phi_2$, which is consistent with the symmetry of v_3 (players 1 and 2 contribute identically to every coalition).

Game v_4

$$v_4(C) = \tag{18.35}$$

1. **Monotonicity of v_4**: We check a few key comparisons.

   - $v_4(\{1\}) = 6 \leq 7 = v_4(\{1,2\})$ ✓
   - $v_4(\{2\}) = 7 \leq 7 = v_4(\{1,2\})$ ✓
   - $v_4(\{1,2\}) = 7 \leq 7 = v_4(\{1,2,3\})$ ✓
   - $v_4(\{1,2,3\}) = 7 \leq 25 = v_4(\{1,2,3,4\})$ ✓
   - $v_4(\{3\}) = 0 \leq 6 = v_4(\{1,3\})$ ✓
   - $v_4(\{1,2,4\}) = 24 \leq 25 = v_4(\{1,2,3,4\})$ ✓

All subset relationships hold. **v_4 is monotone.**

1. **Superadditivity of v_4**: Check a few disjoint pairs.

   - $v_4(\{1,2\}) \geq v_4(\{1\}) + v_4(\{2\})$: $7 \geq 6 + 7 = 13$? No: $7 < 13$.

v_4 is not superadditive.

1. **Shapley value of v_4**: With $N = 4$ there are $4! = 24$ permutations. We use the formula:

$$\phi_i(v_4) = \sum_{S \subseteq N \setminus \{i\}} \frac{|\,S\,|!(|\,N\,| - |\,S\,| - 1)!}{|\,N\,|!} \left[\, v_4(S \cup \{i\}) - v_4(S) \,\right] \tag{18.36}$$

For player 1, the relevant subsets $S \subseteq \{2,3,4\}$ are:

$|\,S\,||\,S\,|$ | weight $= \frac{|\,S\,|!\,(3-|\,S\,|)!}{24}$ | $v_4(S \cup \{1\}) - v_4(S)$ | | – | – | – | – | | $\varnothing$ | 0 | $\frac{0!\cdot 3!}{24} = \frac{6}{24}$ | $v_4(\{1\}) - 0 = 6$ | | $\{2\}$ | 1 | $\frac{1!\cdot 2!}{24} = \frac{2}{24}$ | $v_4(\{1,2\}) - v_4(\{2\}) = 7 - 7 = 0$ | | $\{3\}$ | 1 | $\frac{2}{24}$ | $v_4(\{1,3\}) - v_4(\{3\}) = 6 - 0 = 6$ | | $\{4\}$ | 1 | $\frac{2}{24}$ | $v_4(\{1,4\}) - v_4(\{4\}) = 12 - 8 = 4$ | | $\{2,3\}$ | 2 | $\frac{2!\cdot 1!}{24} = \frac{2}{24}$ | $v_4(\{1,2,3\}) - v_4(\{2,3\}) = 7 - 7 = 0$ | | $\{2,4\}$ | 2 | $\frac{2}{24}$ | $v_4(\{1,2,4\}) - v_4(\{2,4\}) = 24 - 12 = 12$ | | $\{3,4\}$ | 2 | $\frac{2}{24}$ | $v_4(\{1,3,4\}) - v_4(\{3,4\}) = 12 - 8 = 4$ | | $\{2,3,4\}$ | 3 | $\frac{3!\cdot 0!}{24} = \frac{6}{24}$ | $v_4(\{1,2,3,4\}) - v_4(\{2,3,4\}) = 25 - 12 = 13$ |

$$\begin{aligned} \phi_1 &= \frac{1}{24}\left[\, 6\cdot 6 + 2\cdot 0 + 2\cdot 6 + 2\cdot 4 + 2\cdot 0 + 2\cdot 12 + 2\cdot 4 + 6\cdot 13 \,\right] \\ &= \frac{36 + 0 + 12 + 8 + 0 + 24 + 8 + 78}{24} = \frac{166}{24} = \frac{83}{12} \approx 6.92 \end{aligned} \tag{18.37}$$

For player 2, subsets $S \subseteq \{1,3,4\}$:

S	$v_4(S \cup \{2\}) - v_4(S)$
$\varnothing$	7
$\{1\}$	$7 - 6 = 1$
$\{3\}$	$7 - 0 = 7$
$\{4\}$	$12 - 8 = 4$

S	$v_4(S \cup \{2\}) - v_4(S)$
$\{1,3\}$	$7 - 6 = 1$
$\{1,4\}$	$24 - 12 = 12$
$\{3,4\}$	$12 - 8 = 4$
$\{1,3,4\}$	$25 - 12 = 13$

$$\begin{aligned}\phi_2 &= \frac{6\cdot 7 + 2\cdot 1 + 2\cdot 7 + 2\cdot 4 + 2\cdot 1 + 2\cdot 12 + 2\cdot 4 + 6\cdot 13}{24} \\ &= \frac{42 + 2 + 14 + 8 + 2 + 24 + 8 + 78}{24} = \frac{178}{24} = \frac{89}{12} \approx 7.42\end{aligned} \tag{18.38}$$

For player 3, subsets $S \subseteq \{1,2,4\}$:

S	$v_4(S \cup \{3\}) - v_4(S)$
$\varnothing$	0
$\{1\}$	$6 - 6 = 0$
$\{2\}$	$7 - 7 = 0$
$\{4\}$	$8 - 8 = 0$
$\{1,2\}$	$7 - 7 = 0$
$\{1,4\}$	$12 - 12 = 0$
$\{2,4\}$	$12 - 12 = 0$
$\{1,2,4\}$	$25 - 24 = 1$

$$\begin{aligned}\phi_3 &= \frac{6\cdot 0 + 2\cdot 0 + 2\cdot 0 + 2\cdot 0 + 2\cdot 0 + 2\cdot 0 + 2\cdot 0 + 6\cdot 1}{24} \\ &= \frac{6}{24} = \frac{1}{4} = 0.25\end{aligned} \tag{18.39}$$

For player 4, subsets $S \subseteq \{1,2,3\}$:

S	$v_4(S \cup \{4\}) - v_4(S)$
$\varnothing$	8
$\{1\}$	$12 - 6 = 6$
$\{2\}$	$12 - 7 = 5$
$\{3\}$	$8 - 0 = 8$
$\{1,2\}$	$24 - 7 = 17$
$\{1,3\}$	$12 - 6 = 6$
$\{2,3\}$	$12 - 7 = 5$
$\{1,2,3\}$	$25 - 7 = 18$

$$\begin{aligned}\phi_4 &= \frac{6\cdot 8 + 2\cdot 6 + 2\cdot 5 + 2\cdot 8 + 2\cdot 17 + 2\cdot 6 + 2\cdot 5 + 6\cdot 18}{24} \\ &= \frac{48 + 12 + 10 + 16 + 34 + 12 + 10 + 108}{24} = \frac{250}{24} = \frac{125}{12} \approx 10.42\end{aligned} \tag{18.40}$$

Check: $83/12 + 89/12 + 3/12 + 125/12 = 300/12 = 25 = v_4(\{1,2,3,4\})$ ✓

```
import coopgt.characteristic_function_properties
import coopgt.shapley_value

# Game v1
v1 = {
    (): 0,
    (1,): 5, (2,): 3, (3,): 2,
    (1, 2): 12, (1, 3): 5, (2, 3): 4,
    (1, 2, 3): 13,
}
print("v1 monotone:",
coopgt.characteristic_function_properties.is_monotone(v1))
print("v1 superadditive:",
coopgt.characteristic_function_properties.is_superadditive(v1))
print("v1 Shapley:", coopgt.shapley_value.calculate(v1))
```

```
v1 monotone: True
v1 superadditive: False
v1 Shapley: [6.66666667 5.16666667 1.16666667]
```

```
# Game v2
v2 = {(): 0, (1,): 6, (2,): 0, (1, 2): 5}
print("v2 monotone:",
coopgt.characteristic_function_properties.is_monotone(v2))
print("v2 superadditive:",
coopgt.characteristic_function_properties.is_superadditive(v2))
print("v2 Shapley:", coopgt.shapley_value.calculate(v2))
```

```
v2 monotone: False
v2 superadditive: False
v2 Shapley: [ 5.5 -0.5]
```

```
# Game v3
v3 = {
    (): 0,
    (1,): 6, (2,): 6, (3,): 13,
    (1, 2): 6, (1, 3): 13, (2, 3): 13,
    (1, 2, 3): 26,
}
print("v3 monotone:",
coopgt.characteristic_function_properties.is_monotone(v3))
print("v3 superadditive:",
coopgt.characteristic_function_properties.is_superadditive(v3))
print("v3 Shapley:", coopgt.shapley_value.calculate(v3))
```

```
v3 monotone: True
v3 superadditive: False
v3 Shapley: [ 6.33333333  6.33333333 13.33333333]
```

```
# Game v4
v4 = {
    (): 0,
```

```
    (1,): 6, (2,): 7, (3,): 0, (4,): 8,
    (1, 2): 7, (1, 3): 6, (1, 4): 12,
    (2, 3): 7, (2, 4): 12, (3, 4): 8,
    (1, 2, 3): 7, (1, 2, 4): 24, (1, 3, 4): 12, (2, 3, 4): 12,
    (1, 2, 3, 4): 25,
}
print("v4 monotone:",
coopgt.characteristic_function_properties.is_monotone(v4))
print("v4 superadditive:",
coopgt.characteristic_function_properties.is_superadditive(v4))
print("v4 Shapley:", coopgt.shapley_value.calculate(v4))
```

```
v4 monotone: True
v4 superadditive: False
v4 Shapley: [ 6.91666667  7.41666667  0.25       10.41666667]
```

Solution 18.2: Solution to Exercise 2

1. **Characteristic function $v(S)$**

Using the R^2 values from the table, with players $\{x_1, x_2, x_3\}$:

$$v(\varnothing) = 0, \quad v(\{x_1\}) = 0.122, \quad v(\{x_2\}) = 0.097, \quad v(\{x_3\}) = 0.551 \tag{18.41}$$

$$v(\{x_1, x_2\}) = 0.174, \quad v(\{x_1, x_3\}) = 0.581, \quad v(\{x_2, x_3\}) = 0.620 \tag{18.42}$$

$$v(\{x_1, x_2, x_3\}) = 0.623 \tag{18.43}$$

1. **Shapley value computation**

We apply the formula with $N = 3$:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|\,S\,|!(2 - |\,S\,|)!}{6} \left[\, v(S \cup \{i\}) - v(S) \,\right] \tag{18.44}$$

For x_1 (subsets $S \subseteq \{x_2, x_3\}$):

S	weight	$v(S \cup \{x_1\}) - v(S)$
$\varnothing$	$2/6$	$0.122 - 0 = 0.122$
$\{x_2\}$	$1/6$	$0.174 - 0.097 = 0.077$
$\{x_3\}$	$1/6$	$0.581 - 0.551 = 0.030$
$\{x_2, x_3\}$	$2/6$	$0.623 - 0.620 = 0.003$

Here $|\,S\,| = 0$ and $|\,S\,| = 2$ carry weight $\frac{0!\,2!}{6} = \frac{2}{6}$, while $|\,S\,| = 1$ carries weight $\frac{1!\,1!}{6} = \frac{1}{6}$, giving:

$$\phi_{x_1} = \frac{2 \cdot 0.122 + 1 \cdot 0.077 + 1 \cdot 0.030 + 2 \cdot 0.003}{6} = \frac{0.357}{6} \approx 0.0595 \tag{18.45}$$

For x_2 (subsets $S \subseteq \{x_1, x_3\}$):

S	weight	$v(S \cup \{x_2\}) - v(S)$

$\varnothing$	$2/6$	0.097
$\{x_1\}$	$1/6$	$0.174 - 0.122 = 0.052$
$\{x_3\}$	$1/6$	$0.620 - 0.551 = 0.069$
$\{x_1, x_3\}$	$2/6$	$0.623 - 0.581 = 0.042$

$$\phi_{x_2} = \frac{2 \cdot 0.097 + 1 \cdot 0.052 + 1 \cdot 0.069 + 2 \cdot 0.042}{6} = \frac{0.194 + 0.052 + 0.069 + 0.084}{6} = \frac{0.399}{6} \approx 0.0665 \tag{18.46}$$

For x_3 (subsets $S \subseteq \{x_1, x_2\}$):

S	weight	$v(S \cup \{x_3\}) - v(S)$
$\varnothing$	$2/6$	0.551
$\{x_1\}$	$1/6$	$0.581 - 0.122 = 0.459$
$\{x_2\}$	$1/6$	$0.620 - 0.097 = 0.523$
$\{x_1, x_2\}$	$2/6$	$0.623 - 0.174 = 0.449$

$$\phi_{x_3} = \frac{2 \cdot 0.551 + 1 \cdot 0.459 + 1 \cdot 0.523 + 2 \cdot 0.449}{6} = \frac{1.102 + 0.459 + 0.523 + 0.898}{6} = \frac{2.982}{6} \approx 0.4970 \tag{18.47}$$

Check: $0.0595 + 0.0665 + 0.4970 \approx 0.623 = v(\{x_1, x_2, x_3\})$ ✓

1. **Interpretation**

The Shapley values are approximately:

$$\phi_{x_1} \approx 0.060, \quad \phi_{x_2} \approx 0.067, \quad \phi_{x_3} \approx 0.497 \tag{18.48}$$

Feature x_3 contributes by far the most explanatory power, accounting for roughly $0.497/0.623 \approx 80\%$ of the total R^2. Features x_1 and x_2 contribute very little, with x_1 contributing slightly less than x_2. The Shapley value allocates the credit for the combined model performance fairly across the features, accounting for all possible orderings in which they are added.

```
import coopgt.shapley_value

v_linear = {
    (): 0,
    (1,): 0.122,
    (2,): 0.097,
    (3,): 0.551,
    (1, 2): 0.174,
    (1, 3): 0.581,
    (2, 3): 0.620,
    (1, 2, 3): 0.623,
}

shapley = coopgt.shapley_value.calculate(characteristic_function=v_linear)
print("Shapley values (x1, x2, x3):", shapley)
print(f"Sum: {sum(shapley):.3f} (should equal {v_linear[(1, 2, 3)]})")
```

```
Shapley values (x1, x2, x3): [0.0595 0.0665 0.497 ]
Sum: 0.623 (should equal 0.623)
```

Solution 18.3: Solution to Exercise 3

Game v_a on $N = \{1, 2, 3\}$:

$$v_a(C) = \tag{18.49}$$

Game v_b on $N = \{1, 2, 3\}$:

$$v_b(C) = \tag{18.50}$$

1. **Symmetry property**

Recall the symmetry property: if $v(C \cup \{i\}) = v(C \cup \{j\})$ for all $C \subseteq \Omega \setminus \{i, j\}$, then $\phi_i = \phi_j$.

For v_a: Players 1 and 2 are symmetric. For every $C \subseteq \{3\}$:

- $v_a(C \cup \{1\})$: if $C = \varnothing$, $v_a(\{1\}) = 2$; if $C = \{3\}$, $v_a(\{1, 3\}) = 0$.
- $v_a(C \cup \{2\})$: if $C = \varnothing$, $v_a(\{2\}) = 2$; if $C = \{3\}$, $v_a(\{2, 3\}) = 0$.

Since $v_a(C \cup \{1\}) = v_a(C \cup \{2\})$ for all $C \subseteq \{3\}$, players 1 and 2 are symmetric under v_a. No other pair is symmetric, so players 1 and 2 are the only pair the property constrains.

v_a satisfies the symmetry property.

For v_b: Players 1 and 2 are symmetric. For every $C \subseteq \{3\}$:

- $C = \varnothing$: $v_b(\{1\}) = 0 = v_b(\{2\})$ ✓
- $C = \{3\}$: $v_b(\{1, 3\}) = 1 = v_b(\{2, 3\})$ ✓

v_b satisfies the symmetry property.

1. **Shapley values of v_a and v_b**

Shapley value of v_a:

Because v_a is not monotone (the singletons $\{1\}$ and $\{2\}$ are worth 2, but every larger coalition is worth 0), the marginal contributions must be computed carefully, term by term:

π	$\Delta_\pi^{v_a}(1)$	$\Delta_\pi^{v_a}(2)$	$\Delta_\pi^{v_a}(3)$
$(1, 2, 3)$	$v_a(\{1\}) = 2$	$v_a(\{1, 2\}) - v_a(\{1\}) = -2$	$v_a(\{1, 2, 3\}) - v_a(\{1, 2\}) = 0$
$(1, 3, 2)$	2	$v_a(\{1, 2, 3\}) - v_a(\{1, 3\}) = 0$	$v_a(\{1, 3\}) - v_a(\{1\}) = -2$
$(2, 1, 3)$	$v_a(\{1, 2\}) - v_a(\{2\}) = -2$	$v_a(\{2\}) = 2$	0
$(2, 3, 1)$	$v_a(\{1, 2, 3\}) - v_a(\{2, 3\}) = 0$	2	$v_a(\{2, 3\}) - v_a(\{2\}) = -2$
$(3, 1, 2)$	$v_a(\{1, 3\}) - v_a(\{3\}) = 0$	$v_a(\{1, 2, 3\}) - v_a(\{1, 3\}) = 0$	$v_a(\{3\}) = 0$
$(3, 2, 1)$	$v_a(\{1, 2, 3\}) - v_a(\{2, 3\}) = 0$	$v_a(\{2, 3\}) - v_a(\{3\}) = 0$	0

$$\phi_1(v_a) = \frac{2 + 2 - 2 + 0 + 0 + 0}{6} = \frac{2}{6} = \frac{1}{3} \tag{18.51}$$

$$\phi_2(v_a) = \frac{-2+0+2+2+0+0}{6} = \frac{2}{6} = \frac{1}{3} \tag{18.52}$$

$$\phi_3(v_a) = \frac{0-2+0-2+0+0}{6} = \frac{-4}{6} = -\frac{2}{3} \tag{18.53}$$

Check: $1/3 + 1/3 - 2/3 = 0 = v_a(\{1,2,3\})$ ✓

Shapley value of v_b:

π	$\Delta_\pi^{v_b}(1)$	$\Delta_\pi^{v_b}(2)$	$\Delta_\pi^{v_b}(3)$
$(1,2,3)$	0	0	$v_b(\{1,2,3\}) - v_b(\{1,2\}) = 3$
$(1,3,2)$	0	$v_b(\{1,2,3\}) - v_b(\{1,3\}) = 2$	$v_b(\{1,3\}) - v_b(\{1\}) = 1$
$(2,1,3)$	0	0	3
$(2,3,1)$	$v_b(\{1,2,3\}) - v_b(\{2,3\}) = 2$	0	$v_b(\{2,3\}) - v_b(\{2\}) = 1$
$(3,1,2)$	$v_b(\{1,3\}) - v_b(\{3\}) = 1$	$v_b(\{1,2,3\}) - v_b(\{1,3\}) = 2$	0
$(3,2,1)$	$v_b(\{1,2,3\}) - v_b(\{2,3\}) = 2$	$v_b(\{2,3\}) - v_b(\{3\}) = 1$	0

$$\phi_1(v_b) = \frac{0+0+0+2+1+2}{6} = \frac{5}{6} \tag{18.54}$$

$$\phi_2(v_b) = \frac{0+2+0+0+2+1}{6} = \frac{5}{6} \tag{18.55}$$

$$\phi_3(v_b) = \frac{3+1+3+1+0+0}{6} = \frac{8}{6} = \frac{4}{3} \tag{18.56}$$

Check: $5/6 + 5/6 + 4/3 = 5/6 + 5/6 + 8/6 = 18/6 = 3 = v_b(\{1,2,3\})$ ✓

1. **The game $v^+ = v_a + v_b$**

$$v^+(C) = v_a(C) + v_b(C) \tag{18.57}$$

C	$v_a(C)$	$v_b(C)$	$v^+(C)$
$\varnothing$	0	0	0
$\{1\}$	2	0	2
$\{2\}$	2	0	2
$\{3\}$	0	0	0
$\{1,2\}$	0	0	0
$\{1,3\}$	0	1	1
$\{2,3\}$	0	1	1
$\{1,2,3\}$	0	3	3

Using the Shapley formula for v^+:

$$\phi_1(v^+) = \phi_1(v_a) + \phi_1(v_b) = \frac{1}{3} + \frac{5}{6} = \frac{2}{6} + \frac{5}{6} = \frac{7}{6} \tag{18.58}$$

$$\phi_2(v^+) = \frac{1}{3} + \frac{5}{6} = \frac{7}{6} \tag{18.59}$$

$$\phi_3(v^+) = -\frac{2}{3} + \frac{4}{3} = \frac{2}{3} \tag{18.60}$$

1. **Verification of the additivity property**

We verify by computing the Shapley value of v^+ directly.

π	$\Delta_\pi^{v^+}(1)$	$\Delta_\pi^{v^+}(2)$	$\Delta_\pi^{v^+}(3)$
$(1,2,3)$	2	-2	3
$(1,3,2)$	2	2	-1
$(2,1,3)$	-2	2	3
$(2,3,1)$	2	2	-1
$(3,1,2)$	1	2	0
$(3,2,1)$	2	1	0

$$\phi_1(v^+) = \frac{2+2-2+2+1+2}{6} = \frac{7}{6}\checkmark \tag{18.61}$$

$$\phi_2(v^+) = \frac{-2+2+2+2+2+1}{6} = \frac{7}{6}\checkmark \tag{18.62}$$

$$\phi_3(v^+) = \frac{3-1+3-1+0+0}{6} = \frac{4}{6} = \frac{2}{3}\checkmark \tag{18.63}$$

The Shapley value of v^+ equals $\phi(v_a) + \phi(v_b)$, confirming the **additivity property**.

```
import coopgt.shapley_value

va = {
    (): 0,
    (1,): 2, (2,): 2, (3,): 0,
    (1, 2): 0, (1, 3): 0, (2, 3): 0,
    (1, 2, 3): 0,
}

vb = {
    (): 0,
    (1,): 0, (2,): 0, (3,): 0,
    (1, 2): 0, (1, 3): 1, (2, 3): 1,
    (1, 2, 3): 3,
}

v_plus = {k: va[k] + vb[k] for k in va}

phi_a = coopgt.shapley_value.calculate(characteristic_function=va)
phi_b = coopgt.shapley_value.calculate(characteristic_function=vb)
phi_plus = coopgt.shapley_value.calculate(characteristic_function=v_plus)
```

```
print("Shapley value of v_a:", phi_a)
print("Shapley value of v_b:", phi_b)
print("Shapley value of v_a + v_b (direct):", phi_plus)
print("Sum of Shapley values:", [a + b for a, b in zip(phi_a, phi_b)])
print("Additivity holds:", all(abs(p - (a + b)) < 1e-10 for p, a, b in
zip(phi_plus, phi_a, phi_b)))
```

```
Shapley value of v_a: [ 0.33333333  0.33333333 -0.66666667]
Shapley value of v_b: [0.83333333 0.83333333 1.33333333]
Shapley value of v_a + v_b (direct): [1.16666667 1.16666667 0.66666667]
Sum of Shapley values: [np.float64(1.1666666666666667),
np.float64(1.1666666666666667), np.float64(0.6666666666666666)]
Additivity holds: True
```

Solution 18.4: Solution to Exercise 4

The game v on $N = \{1, 2, 3\}$ is:

$$v(C) = \tag{18.64}$$

1. **Identifying null players**

A player i is a null player if $v(C \cup \{i\}) = v(C)$ for all coalitions $C \subseteq \Omega$.

Player 2: Check whether $v(C \cup \{2\}) = v(C)$ for all C:

- $C = \varnothing$: $v(\{2\}) = 0 = v(\varnothing) = 0$ ✓
- $C = \{1\}$: $v(\{1, 2\}) = 7 \neq 4 = v(\{1\})$ ✗

Player 2 is **not** a null player.

Player 3: Check whether $v(C \cup \{3\}) = v(C)$ for all C:

- $C = \varnothing$: $v(\{3\}) = 0 = v(\varnothing) = 0$ ✓
- $C = \{1\}$: $v(\{1, 3\}) = 0 \neq 4 = v(\{1\})$ ✗ (the coalition $\{1, 3\}$ is not listed, so its value is 0).

Player 3 is **not** a null player.

1. **Marginal contributions**

π	$\Delta_\pi^v(1)$	$\Delta_\pi^v(2)$	$\Delta_\pi^v(3)$
$(1, 2, 3)$	$v(\{1\}) - 0 = 4$	$v(\{1, 2\}) - v(\{1\}) = 3$	$v(\{1, 2, 3\}) - v(\{1, 2\}) = 0$
$(1, 3, 2)$	$v(\{1\}) - 0 = 4$	$v(\{1, 2, 3\}) - v(\{1, 3\}) = 7$	$v(\{1, 3\}) - v(\{1\}) = -4$
$(2, 1, 3)$	$v(\{1, 2\}) - v(\{2\}) = 7$	$v(\{2\}) - 0 = 0$	$v(\{1, 2, 3\}) - v(\{1, 2\}) = 0$
$(2, 3, 1)$	$v(\{1, 2, 3\}) - v(\{2, 3\}) = 7$	$v(\{2\}) - 0 = 0$	$v(\{2, 3\}) - v(\{2\}) = 0$
$(3, 1, 2)$	$v(\{1, 3\}) - v(\{3\}) = 0$	$v(\{1, 2, 3\}) - v(\{1, 3\}) = 7$	$v(\{3\}) - 0 = 0$

$(3, 2, 1)$	$v(\{1,2,3\}) - v(\{2,3\}) = 7$	$v(\{2,3\}) - v(\{3\}) = 0$	$v(\{3\}) - 0 = 0$

1. **Shapley value**

$$\phi_1(v) = \frac{4+4+7+7+0+7}{6} = \frac{29}{6} \approx 4.83 \tag{18.65}$$

$$\phi_2(v) = \frac{3+7+0+0+7+0}{6} = \frac{17}{6} \approx 2.83 \tag{18.66}$$

$$\phi_3(v) = \frac{0-4+0+0+0+0}{6} = -\frac{4}{6} = -\frac{2}{3} \approx -0.67 \tag{18.67}$$

Check: $29/6 + 17/6 - 4/6 = 42/6 = 7 = v(\{1,2,3\})$ ✓

Null player property: Player 3 is not a null player (since $v(\{1,3\}) \neq v(\{1\})$), so the null player property does not require $\phi_3 = 0$. Indeed $\phi_3 = -2/3 \neq 0$. The Shapley value correctly penalises player 3 for actually reducing the value of coalition $\{1\}$ when they join.

To see this concretely: adding player 3 to coalition $\{1\}$ reduces the value from 4 to 0. This negative marginal contribution is reflected in the Shapley value.

```
import coopgt.shapley_value

v = {
    (): 0,
    (1,): 4, (2,): 0, (3,): 0,
    (1, 2): 7, (1, 3): 0, (2, 3): 0,
    (1, 2, 3): 7,
}

shapley = coopgt.shapley_value.calculate(characteristic_function=v)
print("Shapley value:", shapley)
print(f"phi_1 = {shapley[0]:.4f} (expected 29/6 = {29/6:.4f})")
print(f"phi_2 = {shapley[1]:.4f} (expected 17/6 = {17/6:.4f})")
print(f"phi_3 = {shapley[2]:.4f} (expected -2/3 = {-2/3:.4f})")
```

```
Shapley value: [ 4.83333333  2.83333333 -0.66666667]
phi_1 = 4.8333 (expected 29/6 = 4.8333)
phi_2 = 2.8333 (expected 17/6 = 2.8333)
phi_3 = -0.6667 (expected -2/3 = -0.6667)
```

Solution 18.5: Solution to Exercise 5

Recall the Shapley value:

$$\phi_i(G) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \Delta_\pi^G(i) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \left[v(S_\pi(i) \cup \{i\}) - v(S_\pi(i)) \right] \tag{18.68}$$

Proof of Efficiency

We must show $\sum_{i=1}^{N} \phi_i(G) = v(\Omega)$.

$$\sum_{i=1}^{N} \phi_i(G) = \frac{1}{N!} \sum_{i=1}^{N} \sum_{\pi \in \Pi_N} \Delta_\pi^G(i) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \sum_{i=1}^{N} \Delta_\pi^G(i) \tag{18.69}$$

For a fixed permutation $\pi = (\pi_1, \pi_2, ..., \pi_N)$, the marginal contributions telescope:

$$\sum_{i=1}^{N} \Delta_\pi^G(i) = \sum_{k=1}^{N} \left[\, v(\{\pi_1, ..., \pi_k\}) - v(\{\pi_1, ..., \pi_{k-1}\}) \, \right] = v(\Omega) - v(\varnothing) = v(\Omega) \tag{18.70}$$

since $v(\varnothing) = 0$. Therefore:

$$\sum_{i=1}^{N} \phi_i(G) = \frac{1}{N!} \sum_{\pi \in \Pi_N} v(\Omega) = \frac{N! \cdot v(\Omega)}{N!} = v(\Omega) \qquad \square \tag{18.71}$$

Proof of the Null Player Property

Suppose $v(C \cup \{i\}) = v(C)$ for all $C \subseteq \Omega$. Then for every permutation π:

$$\Delta_\pi^G(i) = v(S_\pi(i) \cup \{i\}) - v(S_\pi(i)) = 0 \tag{18.72}$$

Therefore:

$$\phi_i(G) = \frac{1}{N!} \sum_{\pi \in \Pi_N} 0 = 0 \qquad \square \tag{18.73}$$

Proof of the Symmetry Property

Suppose $v(C \cup \{i\}) = v(C \cup \{j\})$ for all $C \subseteq \Omega \setminus \{i, j\}$. We wish to show $\phi_i = \phi_j$.

For any permutation π, define the permutation π' by swapping the positions of i and j in π. This defines a bijection on Π_N (swapping i and j in every permutation), and every permutation is paired with a unique partner.

Consider any permutation π. Let $S = S_\pi(i) \setminus \{j\}$, i.e. the predecessors of i not counting j.

Case 1: j is not a predecessor of i in π (i.e. i comes before j). Then $S_\pi(i) \subseteq \Omega \setminus \{i, j\}$, so:

$$\Delta_\pi^G(i) = v(S_\pi(i) \cup \{i\}) - v(S_\pi(i)) \tag{18.74}$$

In π' (where i and j are swapped), j now appears where i was, so $S_{\pi'}(j) = S_\pi(i)$:

$$\Delta_{\pi'}^G(j) = v(S_{\pi'}(j) \cup \{j\}) - v(S_{\pi'}(j)) = v(S_\pi(i) \cup \{j\}) - v(S_\pi(i)) \tag{18.75}$$

Since $S_\pi(i) \subseteq \Omega \setminus \{i, j\}$, the symmetry assumption gives $v(S_\pi(i) \cup \{i\}) = v(S_\pi(i) \cup \{j\})$, so:

$$\Delta_\pi^G(i) = \Delta_{\pi'}^G(j) \tag{18.76}$$

Case 2: j is a predecessor of i in π. Then $S_\pi(i)$ contains j. Let $S = S_\pi(i) \setminus \{j\} \subseteq \Omega \setminus \{i, j\}$. Then:

$$\Delta_\pi^G(i) = v(S \cup \{i, j\}) - v(S \cup \{j\}) \tag{18.77}$$

In π', i has been moved to where j was, so the predecessors of j in π' are the predecessors of i in π with j relabelled as i, namely $S \cup \{i\}$. Thus:

$$\Delta_{\pi'}^G(j) = v(S \cup \{i, j\}) - v(S \cup \{i\}) \tag{18.78}$$

Since $S \subseteq \Omega \setminus \{i, j\}$, the symmetry assumption gives $v(S \cup \{i\}) = v(S \cup \{j\})$, so $\Delta_\pi^G(i) = \Delta_{\pi'}^G(j)$ in this case as well.

In both cases $\Delta_{\pi}^{G}(i) = \Delta_{\pi'}^{G}(j)$. Since $\pi \mapsto \pi'$ is a bijection on Π_N, summing over all permutations gives:

$$\phi_i(G) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \Delta_{\pi}^{G}(i) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \Delta_{\pi'}^{G}(j) = \frac{1}{N!} \sum_{\pi' \in \Pi_N} \Delta_{\pi'}^{G}(j) = \phi_j(G) \qquad \square \tag{18.79}$$

Proof of the Additivity Property

Let $G_1 = (N, v_1)$, $G_2 = (N, v_2)$, and $G^{+} = (N, v^{+})$ with $v^{+} = v_1 + v_2$.

$$\phi_i(G^{+}) = \frac{1}{N!} \sum_{\pi \in \Pi_N} \left[\, v^{+}(S_\pi(i) \cup \{i\}) - v^{+}(S_\pi(i)) \, \right] \tag{18.80}$$

$$= \frac{1}{N!} \sum_{\pi \in \Pi_N} \left[\, (v_1 + v_2)(S_\pi(i) \cup \{i\}) - (v_1 + v_2)(S_\pi(i)) \, \right] \tag{18.81}$$

$$= \frac{1}{N!} \sum_{\pi \in \Pi_N} \left[\, v_1(S_\pi(i) \cup \{i\}) - v_1(S_\pi(i)) \, \right] + \frac{1}{N!} \sum_{\pi \in \Pi_N} \left[\, v_2(S_\pi(i) \cup \{i\}) - v_2(S_\pi(i)) \, \right] \tag{18.82}$$

$$= \phi_i(G_1) + \phi_i(G_2) \qquad \square \tag{18.83}$$

19. The Core

The Shapley value asks how to divide the value of a coalition *fairly*. It says nothing about whether the players will accept that division. This chapter studies the **core**, the set of allocations that are *stable* in the sense that no group of players can do better by breaking away. Fairness and stability are different demands, and as we will see they need not agree.

19.1 Motivating Example: Will the party stick together?

Recall the D&D battle from the previous chapter, in which Ren, Azel, and Quinn divide 1,000 gold coins. The Shapley value divides the hoard fairly, giving

$$\phi = \left(\phi_{\text{Ren}}, \phi_{\text{Azel}}, \phi_{\text{Quinn}}\right) = (350, 450, 200). \tag{19.1}$$

This is a fair split: each character receives their average marginal contribution. But is it stable? Recall that Ren and Azel together can secure 900 coins on their own, since $v(\{\text{Ren}, \text{Azel}\}) = 900$. Under the Shapley value they receive only $350 + 450 = 800$ between them. The two of them would do better to walk away from Quinn and split the 900 they can earn alone.

The fair allocation is therefore not stable: a sub-group is being short-changed relative to what it could guarantee itself. If the party is to stay together, the division must give every coalition at least what it could secure on its own. The set of allocations meeting that demand is the subject of this chapter.

19.2 Theory

19.2.1 Definition: Imputation

Given a characteristic function game $G = (N, v)$, a **payoff vector** $x \in \mathbb{R}^N$ is an **imputation** if it is:

- **efficient**: $\sum_{i \in N} x_i = v(N)$, the whole value of the grand coalition is shared out; and
- **individually rational**: $x_i \geq v(\{i\})$ for every player i, so that no player receives less than they could secure alone.

Individual rationality is the weakest possible stability requirement: it asks only that no *single* player wishes to leave. The core strengthens this to every coalition.

19.2.1.1 Example: Imputations of the D&D battle

For the D&D battle the grand coalition is worth $v(N) = 1000$ and each player alone is worth $v(\{i\}) = 0$, so a payoff vector is an imputation precisely when it sums to 1000 and gives every player a non-negative amount. The Shapley value $(350, 450, 200)$ is an imputation, and so is the lopsided split $(1000, 0, 0)$ in which Ren takes everything. The vector $(1100, -100, 0)$ is **not** an imputation: it is efficient, but Azel receives $-100 < v(\{\text{Azel}\}) = 0$, less than they could secure alone. Individual rationality alone permits a great many allocations; the core will cut them down.

19.2.2 Definition: The Core

The **core** of a characteristic function game $G = (N, v)$ is the set of efficient payoff vectors on which no coalition can improve:

$$\mathcal{C}(G) = \left\{ x \in \mathbb{R}^N \mid \sum_{i \in N} x_i = v(N) \quad \text{and} \quad \sum_{i \in C} x_i \geq v(C) \quad \text{for all } C \subseteq N \right\}. \tag{19.2}$$

The defining inequalities are the **coalitional rationality** constraints: every coalition C must receive, in total, at least the value $v(C)$ it could secure by itself. A coalition for which $\sum_{i \in C} x_i < v(C)$ is said to **block** the allocation x; the core is exactly the set of efficient, unblocked allocations. Since the singleton constraints $x_i \geq v(\{i\})$ are among the coalitional ones, every core allocation is an imputation, but the converse need not hold.

> **Note**
>
> The core is a **polytope**: it is the set of points in $\mathbb{R}^N$ satisfying one linear equality and a finite collection of linear inequalities, one per coalition. It may be empty, a single point, or a higher-dimensional region. This is the same polytope structure studied in Best response polytopes, and as we will see it makes core membership a question of linear programming.

19.2.2.1 Example: The core of the D&D battle

For the D&D battle, an allocation $x = (x_{\text{Ren}}, x_{\text{Azel}}, x_{\text{Quinn}})$ is in the core if it is efficient, $x_{\text{Ren}} + x_{\text{Azel}} + x_{\text{Quinn}} = 1000$, and no coalition can block it:

$$\begin{aligned} x_{\text{Ren}} + x_{\text{Azel}} &\geq 900, \\ x_{\text{Ren}} + x_{\text{Quinn}} &\geq 400, \\ x_{\text{Azel}} + x_{\text{Quinn}} &\geq 600, \\ x_{\text{Ren}}, x_{\text{Azel}}, x_{\text{Quinn}} &\geq 0. \end{aligned} \tag{19.3}$$

The Shapley value $(350, 450, 200)$ violates the first inequality, since $350 + 450 = 800 < 900$, confirming that it is not in the core. A stable allocation must give the strong pair $\{\text{Ren}, \text{Azel}\}$ at least 900 between them. The allocation $(350, 600, 50)$ satisfies every inequality ($950 \geq 900$, $400 \geq 400$, $650 \geq 600$) and so lies in the core: the party will stay together under it. The core here is not empty, but the fair allocation lies outside it.

19.2.3 Definition: Convex game

A characteristic function game $G = (N, v)$ is **convex** (or supermodular) if

$$v(S \cup T) + v(S \cap T) \geq v(S) + v(T) \quad \text{for all coalitions } S, T \subseteq N. \tag{19.4}$$

Equivalently, a game is convex when the marginal contribution of a player to a coalition never decreases as the coalition grows: players become more valuable as more of the others join. Convexity is stronger than superadditivity, which is the special case where S and T are disjoint.

19.2.3.1 Example: The D&D battle is not convex

In the D&D battle, adding Quinn to the coalition $\{\text{Ren}\}$ raises its value by $v(\{\text{Ren}, \text{Quinn}\}) - v(\{\text{Ren}\}) = 400$, while adding Quinn to the larger coalition $\{\text{Ren}, \text{Azel}\}$ raises it by only $v(N) - v(\{\text{Ren}, \text{Azel}\}) = 100$. Quinn's marginal contribution *falls* as the coalition grows, so the game is not convex. This is exactly why, as the next theorem makes precise, its fair and stable allocations can come apart.

19.2.4 Theorem: Convex games have a non-empty core

If $G = (N, v)$ is convex, then its core is non-empty and contains the Shapley value.

We state this result without proof; it is due to Shapley (Shapley, 1971). Convexity is therefore a convenient sufficient condition: for a convex game the fair allocation and a stable allocation coincide,

because the Shapley value is itself in the core. The D&D battle shows that without convexity the two can come apart: it is not convex, as the example above shows, and its Shapley value lies outside its core.

19.2.5 Theorem: The Bondareva–Shapley theorem

The core may be empty, and there is an exact condition for when it is not. A collection of weights $(\lambda_C)_{C \subseteq N,\, C \neq \varnothing}$ with $\lambda_C \geq 0$ is **balanced** if, for every player i, $\sum_{C\,:\,i \in C} \lambda_C = 1$. The game is **balanced** if for every such balanced collection $\sum_C \lambda_C v(C) \leq v(N)$.

The core of $G = (N, v)$ is non-empty if and only if the game is balanced.

We state the theorem, due independently to Bondareva (Bondareva, 1963) and Shapley (Shapley, 1967), without proof. Its practical content is that core non-emptiness is a **linear programming** question. The core is non-empty precisely when the linear program

$$\text{minimise} \sum_{i \in N} x_i \quad \text{subject to} \quad \sum_{i \in C} x_i \geq v(C) \quad \text{for all } C \neq \varnothing \tag{19.5}$$

has optimal value equal to $v(N)$; the balancedness condition is exactly the statement that the dual of this program never exceeds $v(N)$. When the optimum equals $v(N)$, the optimal x is itself a point of the core.

> **Note**
>
> An empty core is a strong statement about a game: it means that *no* division of $v(N)$ is stable. Whatever allocation is proposed, some coalition can guarantee its members strictly more than they are offered and will break away, so the grand coalition cannot be held together by any binding agreement. The three-player majority game of Exercise 2 is the classic example: every split of the surplus is overturned by some two-player majority, and the bargaining never settles.

19.2.6 Definition: The nucleolus

When the core is non-empty it is usually not a single point, and we may want to select one allocation from it. For a payoff vector x and coalition C, the **excess** $e(C, x) = v(C) - \sum_{i \in C} x_i$ measures how dissatisfied C is with x: a positive excess means C is blocking. The **nucleolus** is the imputation that lexicographically minimises the vector of excesses sorted from largest to smallest, that is, it first makes the most dissatisfied coalition as satisfied as possible, then the next, and so on.

The nucleolus always exists, is unique, and whenever the core is non-empty it lies in the core. It can be read as the most stable allocation of all: it pushes every coalition as far from blocking as the game allows.

19.2.6.1 Example: The nucleolus of the D&D battle

For the D&D battle the singleton coalitions have excess $e(\{i\}, x) = -x_i \leq 0$ and so are never the most dissatisfied; the binding coalitions are the three pairs. Using efficiency $x_{\text{Ren}} + x_{\text{Azel}} + x_{\text{Quinn}} = 1000$, their excesses are

$$\begin{aligned} e(\{\text{Ren}, \text{Azel}\}, x) &= 900 - (x_{\text{Ren}} + x_{\text{Azel}}) = x_{\text{Quinn}} - 100, \\ e(\{\text{Ren}, \text{Quinn}\}, x) &= 400 - (x_{\text{Ren}} + x_{\text{Quinn}}) = x_{\text{Azel}} - 600, \\ e(\{\text{Azel}, \text{Quinn}\}, x) &= 600 - (x_{\text{Azel}} + x_{\text{Quinn}}) = x_{\text{Ren}} - 400. \end{aligned} \tag{19.6}$$

These three excesses always sum to $1000 - 1100 = -100$, so the largest of them is at least the average $-100/3$, with equality only when all three are equal. The nucleolus makes the largest excess as small as possible, and so equalises them, giving

$$x = \left(\frac{1100}{3}, \frac{1700}{3}, \frac{200}{3} \right) \approx (366.7, 566.7, 66.7). \tag{19.7}$$

This allocation lies in the core, and it differs from both the Shapley value $(350, 450, 200)$ and the core allocation $(350, 600, 50)$ met earlier: the nucleolus is the single division that holds every coalition as far from blocking as the game permits.

19.3 Exercises

Exercise 19.1:

Consider the three-player game with

$$v(C) = \tag{19.8}$$

1. Show that the allocation $(3, 3, 3)$ is an imputation.
2. Determine whether $(3, 3, 3)$ is in the core. If not, identify a blocking coalition.
3. Find an allocation that is in the core.

Exercise 19.2:

Consider the three-player majority game, in which any coalition of two or more players can secure the whole value:

$$v(C) = \tag{19.9}$$

Prove that the core is empty.

Exercise 19.3:

Consider the game $v(C) = |\ C\ |^2$ on three players.

1. Verify that the game is convex.
2. Compute the Shapley value.
3. Confirm that the Shapley value lies in the core.

Exercise 19.4:

In a glove market, players 1 and 2 each own a left glove and player 3 owns a right glove. A matched pair of gloves is worth 1, so the value of a coalition is the number of complete pairs it can form:

$$v(C) = \min(\ell, r), \tag{19.10}$$

where ℓ and r are the numbers of left and right gloves held by the members of C.

1. Write down the characteristic function.
2. Find the core, and interpret the result.

19.4 Programming

The `coopgt` library can check whether a payoff vector is an imputation or lies in the core, and whether a game is convex. We continue with the D&D battle.

```
import coopgt.core
import coopgt.shapley_value

characteristic_function = {
    (): 0,
    (1,): 0,
    (2,): 0,
    (3,): 0,
    (1, 2): 900,
    (1, 3): 400,
    (2, 3): 600,
    (1, 2, 3): 1000,
}

shapley = coopgt.shapley_value.calculate(characteristic_function)
print(f"Shapley value: {shapley}")
```

```
Shapley value: [350. 450. 200.]
```

The Shapley value is fair, but it is not in the core, because the coalition $\{1, 2\}$ can block it:

```
print(f"Shapley value in the core?
{coopgt.core.is_in_core(characteristic_function, shapley)}")
```

```
Shapley value in the core? False
```

The game is not convex, which is why fairness and stability come apart:

```
print(f"Game convex? {coopgt.core.is_convex(characteristic_function)}")
```

```
Game convex? False
```

By the Bondareva–Shapley theorem the core is non-empty exactly when the game is balanced. `coopgt` finds a point of the core by solving the associated linear program, returning `None` when the core is empty:

```
core_point = coopgt.core.find_core_point(characteristic_function)
print(f"A point in the core: {core_point}")
```

```
A point in the core: [400. 600.  -0.]
```

An allocation is returned rather than `None`, so the core is non-empty, and we can confirm that the allocation it found is indeed a point of the core:

```
print(f"Found allocation in the core?
{coopgt.core.is_in_core(characteristic_function, core_point)}")
```

```
Found allocation in the core? True
```

Note

The core-membership functions used here are part of `coopgt` from version `0.0.3`. Earlier versions provide only the Shapley value and the characteristic function property checks.

19.5 Notable Research

The core was introduced by Gillies (Gillies, 1959) as part of his study of stable allocations in cooperative games, building on the stable-set ideas of von Neumann and Morgenstern. The exact condition for the core to be non-empty was established independently by Bondareva (Bondareva, 1963) and Shapley (Shapley, 1967) in the balancedness theorem that now bears both their names, tying core non-emptiness to linear programming duality.

Shapley (Shapley, 1971) showed that convex games always have a non-empty core whose extreme points are the marginal-contribution vectors, and that the Shapley value is their barycentre and hence always in the core. The nucleolus was introduced by Schmeidler (Schmeidler, 1969), who proved that it always exists, is unique, and lies in the core whenever the core is non-empty.

Cooperative solution concepts of this kind underpin practical cost-sharing and resource-allocation schemes, from apportioning the cost of shared infrastructure to allocating gains in collaborative logistics, where stability is what keeps a coalition of firms or municipalities from walking away.

19.6 Conclusion

In this chapter we studied the **core**, the set of allocations that no coalition can improve upon. Where the Shapley value answers the question of *fairness*, the core answers the question of *stability*, and the D&D battle shows that the two can disagree: the fair allocation left the strong pair worse off than they could be alone.

We saw that the core is a polytope, that it can be empty, and that the **Bondareva–Shapley theorem** characterises non-emptiness through balancedness, turning the question into a linear program. **Convex games** are a well-behaved class for which the core is always non-empty and contains the Shapley value, so that fairness and stability coincide. Finally the **nucleolus** selects a single, maximally stable allocation from the core whenever one exists.

Table 19.1 summarises the concepts of this chapter.

Concept	Description
Imputation	An efficient, individually rational payoff vector
Core	The efficient allocations no coalition can block
Blocking coalition	A coalition C with $\sum_{i \in C} x_i < v(C)$
Convex game	A game with $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$
Bondareva–Shapley theorem	The core is non-empty if and only if the game is balanced
Nucleolus	The unique imputation lexicographically minimising excesses

Table 19.1: Summary of core concepts

Important

The core is the set of efficient allocations that are stable against every coalition. It can be empty; the Bondareva–Shapley theorem says it is non-empty exactly when the game is balanced, and convex games are always balanced.

19.7 Solutions

Solution 19.1: Solution to Exercise 1

1. The allocation $(3,3,3)$ is efficient, since $3+3+3=9=v(\{1,2,3\})$, and individually rational, since each player receives $3\geq 0=v(\{i\})$. It is therefore an imputation.
2. The coalition $\{1,2\}$ receives $3+3=6$, but $v(\{1,2\})=7$. Since $6<7$, the coalition $\{1,2\}$ blocks the allocation, so $(3,3,3)$ is **not** in the core.
3. A core allocation must satisfy $x_1+x_2\geq 7$, $x_1+x_3\geq 5$, $x_2+x_3\geq 5$, and $x_1+x_2+x_3=9$. The first condition forces $x_3\leq 2$, and the other two force $x_1\geq 3$ and $x_2\geq 3$. The allocation $(3.5,3.5,2)$ satisfies all of them: $7\geq 7$, $5.5\geq 5$, and $5.5\geq 5$. It is therefore in the core.

Solution 19.2: Solution to Exercise 2

Suppose, for contradiction, that $x=(x_1,x_2,x_3)$ is in the core. Then each two-player coalition is unblocked:

$$x_1+x_2\geq 1, \qquad x_1+x_3\geq 1, \qquad x_2+x_3\geq 1. \tag{19.11}$$

Adding these three inequalities gives $2(x_1+x_2+x_3)\geq 3$, that is, $x_1+x_2+x_3\geq 3/2$. But efficiency requires $x_1+x_2+x_3=v(N)=1$, and $1<3/2$. This is a contradiction, so no such x exists and the core is empty. There is therefore no stable way to split the unit of value: whichever two players agree to share it, the third can always tempt one of them into a new majority, and the bargaining cycles without end.

Solution 19.3: Solution to Exercise 3

1. With $v(C)=|\,C\,|^2$, write $a=|\,S\,|$, $b=|\,T\,|$, $u=|\,S\cap T\,|$, so that $|\,S\cup T\,|=a+b-u$. Convexity requires $(a+b-u)^2+u^2\geq a^2+b^2$. Expanding the left-hand side gives $a^2+b^2+2u^2+2ab-2au-2bu=a^2+b^2+2(a-u)(b-u)$. Since $u=|\,S\cap T\,|\leq \min(|\,S\,|,|\,T\,|)$, both $a-u\geq 0$ and $b-u\geq 0$, so the extra term is non-negative and the inequality holds. The game is convex.
2. By symmetry the three players are interchangeable, so the Shapley value splits $v(N)=9$ equally: $\phi=(3,3,3)$.
3. Each singleton receives $3\geq 1=v(\{i\})$, each pair receives $6\geq 4=v(\{i,j\})$, and the grand coalition receives $9=v(N)$. No coalition can block, so $(3,3,3)$ is in the core, as guaranteed by the theorem on convex games.

Solution 19.4: Solution to Exercise 4

1. Counting complete pairs in each coalition gives

$$v(C)= \tag{19.12}$$

The coalition $\{1,2\}$ holds two left gloves and no right glove, so $v(\{1,2\})=0$.

1. A core allocation satisfies $x_1+x_3\geq 1$, $x_2+x_3\geq 1$, and $x_1+x_2+x_3=1$ with $x_i\geq 0$. Substituting $x_1=1-x_2-x_3$ into $x_1+x_3\geq 1$ gives $1-x_2\geq 1$, so $x_2\leq 0$ and hence

$x_2 = 0$; by the same argument with players 1 and 2 exchanged, $x_1 = 0$. Efficiency then forces $x_3 = 1$. The core is the single point $(0, 0, 1)$.

The scarce right glove captures the entire value. Although players 1 and 2 own two thirds of the gloves, their gloves are in surplus, so stability awards them nothing. The core makes the value of scarcity precise.

20. Appendix: Numerical Integration

20.1 Motivating Example: Air resistance of a sky diver

An object falling under gravity is often assumed to not be subject to air resistance. This gives:

$$\frac{dv}{dt} = g \tag{A}$$

This differential equation can be solved to give:

$$v(t) = -gt + v_0 \tag{B}$$

where $v_0 = v(0)$. This in turn gives the vertical displacement of an object falling as:

$$\frac{dy}{dt} = v(t) = -gt + v_0 \tag{C}$$

which can be solved to give:

$$y(t) = -g\frac{t^2}{2} + v_0 t + y_0 \tag{D}$$

where $y(0) = y_0$. This is all based on (20.1) which assumes no air resistance. In fact all objects falling experience a drag coefficient k (kg/m). Approximate values of k are given in Table 20.1.

Position	k (kg/m)
Spread-eagle	0.42875
Aerodynamic	0.07718
Parachute	26.79688

Table A: Estimated quadratic drag coefficients for skydiver

To take this coefficient in to account (20.1) is modified to give:

$$\frac{dv}{dt} = g - \frac{k}{m}v^2 \tag{E}$$

This equation can in fact not be solved to give a closed form solution. Another approach to be able to compute the trajectory of a skydiver is needed. The approach in question is numerical integration and the specific approach in this appendix is Euler's method.

20.2 Theory

20.2.1 Definition: Euler's Method

Euler's Method is a first-order numerical procedure for approximating solutions to initial value problems of the form:

$$\frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0 \tag{F}$$

Given a step size $h > 0$, Euler's Method generates a sequence of approximations $\{(t_n, y_n)\}$ by:

$$t_{n+1} = t_n + h \tag{G}$$

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{H}$$

starting from the initial condition $y(t_0) = y_0$.

At each step, Euler's Method uses the slope given by the differential equation at the current point to extrapolate forward by a small time step.

The differential equation defines a slope field in the (t, y) plane. Euler's Method approximates the solution curve by connecting short tangent segments. At each step, the next point is found by moving forward along the tangent line to the curve at the current point.

This can be visualised as constructing a polygonal approximation to the true solution curve as shown in Figure 20.1.

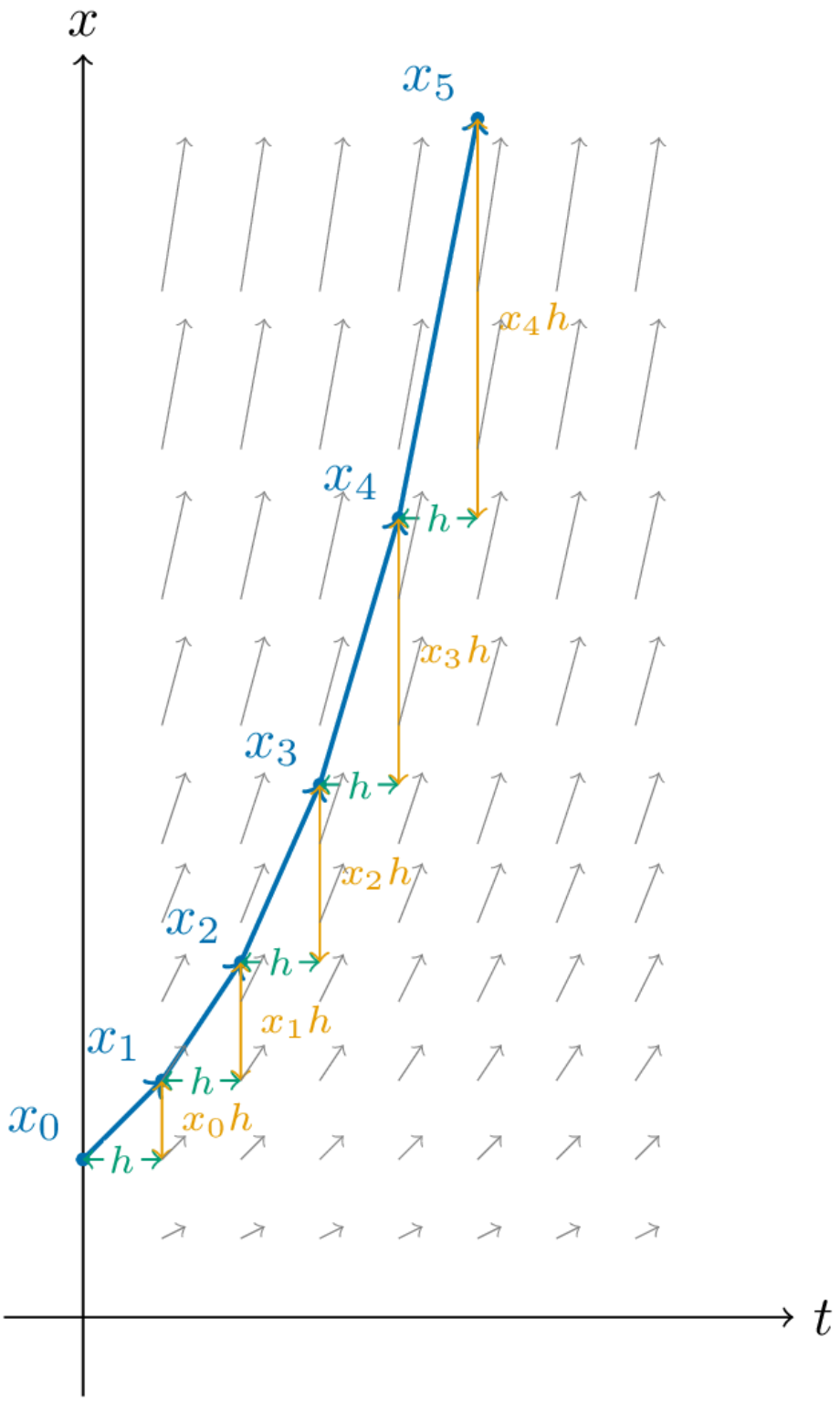


Figure A: A geometric interpretation of numerical integration applied to $\frac{dx}{dt} = x$.

Attention

Euler's Method is **first-order accurate**, meaning the global error scales linearly with the step size h:

$$\text{Global error} = \mathcal{O}(h) \tag{I}$$

Although simple, Euler's Method is only conditionally stable and may require very small step sizes for good accuracy. More accurate alternatives include Runge-Kutta methods, which use higher-order derivative information.

20.2.2 Example: Exponential growth

Consider the differential equation:

$$\frac{dx}{dt} = x, \qquad x(0) = 1 \tag{J}$$

The exact solution is $x(t) = e^t$.

Applying Euler's Method with step size h gives:

$$x_{n+1} = x_n + hx_n = x_n(1 + h) \tag{K}$$

which generates the sequence:

$$x_n = (1 + h)^n \tag{L}$$

As $h \to 0$ and $n \to \infty$ with $nh = t$, this approximates e^t.

20.3 Exercises

Exercise A:

Use Euler's method to approximate the solution to the differential equation:

$$\frac{dx}{dt} = x, \qquad x(0) = 1 \tag{M}$$

with step size $h = 0.2$ up to $t = 1$.

1. Compute the values of $x_1, x_2, \ldots, x_5$ using Euler's method.
2. Compare each value to the exact solution $x(t) = e^t$ at the same time points.
3. Comment on how the approximation behaves: is it an overestimate or an underestimate? Why?

Exercise B:

Consider the differential equation:

$$\frac{dx}{dt} = -2x + 1, \qquad x(0) = 0 \tag{N}$$

1. Use Euler's method with step size $h = 0.1$ to compute the first 5 steps.
2. Plot the approximation.
3. Sketch or describe the qualitative behaviour of the true solution.

Exercise C:

Let $x(t)$ be the true solution to the initial value problem:

$$\frac{dx}{dt} = \cos(t), \qquad x(0) = 0 \tag{O}$$

1. Derive the exact solution.
2. Use Euler's method with step sizes $h = 0.5$, 0.25, and 0.125 to compute $x(1)$.
3. For each step size, compute the absolute error at $t = 1$.
4. Discuss how the error changes as h decreases. What does this suggest about the convergence of Euler's method?

20.4 Programming

20.4.1 Writing an implementation of Euler's method

The following python function implements Euler's method:

```
def get_euler_steps(function, number_of_steps, h, x_0):
    """
    Given a differential equation of the following form:

    dx/dt = function(x)

    This returns `number_of_steps` as given by Euler's algorithm:

    x_{n + 1} = x_n + h function(x_n)

    Parameters
    ----------
    function : A python function -- corresponding to the right hand side of the
               ordinary differential equation
    number_of_steps : int -- The number of iterations of the algorithm
    h : float -- The step size
    x_0 : float -- the initial_value of x

    Returns
    -------
    steps : A python list
    """
    steps = [x_0]
    for _ in range(number_of_steps):
        steps.append(steps[-1] + h * function(steps[-1]))
    return steps
```

We can use this function to get the steps as required:

```
def derivative(x):
    return x

number_of_steps = 10
h = 0.1
x_0 = 1
steps = get_euler_steps(
```

```
    function=derivative,
    number_of_steps=number_of_steps,
    h=h,
    x_0=x_0,
)
steps
```

```
[1,
 1.1,
 1.2100000000000002,
 1.3310000000000002,
 1.4641000000000002,
 1.61051,
 1.7715610000000002,
 1.9487171,
 2.1435888100000002,
 2.357947691,
 2.5937424601]
```

20.4.2 Using Scipy's ordinary differential equation integrator

The Scipy library has an implementation of numerical integration for ordinary differential equations based on an algorithm described in (Petzold, 1983). One difference with the previous section is that the function corresponding to the right hand side of the ordinary differential equation must take t as an input even if it does not use it and it also takes a sequence of time points instead of a number of steps. Let us start by setting those up:

```
import scipy.integrate
import numpy as np

def derivative(x, t):
    return x

t = np.linspace(0, 1, 10)
```

```
steps = scipy.integrate.odeint(func=derivative, t=t, y0=x_0)
steps
```

```
array([[1.        ],
       [1.11751906],
       [1.24884886],
       [1.39561243],
       [1.55962349],
       [1.742909  ],
       [1.94773405],
       [2.17663003],
       [2.43242551],
       [2.7182819 ]])
```

Attention

When using numerical integration it is important to carefully consider the step size (or the number of points) to ensure accuracy.

20.5 Notable Research

The earliest method of numerical integration, now known as **Euler's method**, originates from the foundational work of Leonhard Euler in the 18th century. He implicitly described the approach in (Euler, 1768), laying the groundwork for using tangent approximations to solve differential equations numerically.

The natural development from Euler's idea led to the class of **Runge–Kutta methods**, which provide higher-order accuracy by sampling the derivative at multiple points. These were introduced independently by Runge (Runge, 1895) and Kutta (Kutta, 1901), and remain the basis of many modern solvers.

In contemporary settings, attention has turned to the distinction between **stiff** and **non-stiff** problems. These require different integration techniques for efficient and stable solutions. Foundational references include (Hairer et al., 1993) for non-stiff problems and (Hairer & Wanner, 1996) for stiff and differential-algebraic systems.

An important algorithm widely used today is **LSODA**, described in (Petzold, 1983). This method automatically switches between stiff and non-stiff solvers based on the behaviour of the system, and it underpins many solvers, including those in the SciPy library.

20.6 Conclusion

Euler's method offers a simple and intuitive introduction to numerical integration, but it highlights the tradeoff between simplicity, accuracy, and stability. Modern applications often involve stiff systems or demand high accuracy, where more advanced methods are preferable.

Table 20.2 summarises key properties of several commonly used numerical integration methods.

Method	Order of Accuracy	Stiff Solver	Notes
Euler	1	No	Simple, intuitive
RK4	4	No	Widely used, good for smooth ODEs
Backward Euler	1	Yes	Implicit, stable for stiff systems
LSODA	Adaptive	Yes	Switches between stiff/non-stiff

Table B: Summary of numerical methods. Higher-order methods reduce error more quickly with smaller steps. Stiff solvers handle problems where some parts evolve much faster than others.

Important

Euler's method is valuable for learning, but for practical use, one should default to a well-tested numerical integrator unless there's a compelling reason not to.

20.7 Solutions

Solution A: Solution to Exercise A1.1

The differential equation is

$$\frac{dx}{dt} = x, \qquad x(0) = 1, \tag{P}$$

with step size $h = 0.2$. Euler's method gives

$$x_{n+1} = x_n + h \cdot x_n = x_n(1+h) = 1.2\, x_n. \tag{Q}$$

1. Starting from $x_0 = 1$, the five Euler steps are:

$$\begin{aligned} x_1 &= 1.2^1 = 1.2000 \\ x_2 &= 1.2^2 = 1.4400 \\ x_3 &= 1.2^3 = 1.7280 \\ x_4 &= 1.2^4 = 2.0736 \\ x_5 &= 1.2^5 = 2.4883 \end{aligned} \tag{R}$$

2. The exact solution is $x(t) = e^t$. Comparing at $t_n = 0.2n$:

n	t_n	x_n (Euler)	e^{t_n} (exact)	Absolute error
1	0.2	1.2000	1.2214	0.0214
2	0.4	1.4400	1.4918	0.0518
3	0.6	1.7280	1.8221	0.0941
4	0.8	2.0736	2.2255	0.1519
5	1.0	2.4883	2.7183	0.2300

```python
import numpy as np

h = 0.2
x = 1.0
print(f"{'n':>3}  {'t':>5}  {'Euler':>10}  {'Exact':>10}  {'Error':>10}")
for n in range(1, 6):
    x = x * (1 + h)
    t = n * h
    exact = np.exp(t)
    print(f"{n:>3}  {t:>5.1f}  {x:>10.6f}  {exact:>10.6f}  {abs(x - exact):>10.6f}")
```

```
  n      t       Euler       Exact       Error
  1    0.2    1.200000    1.221403    0.021403
  2    0.4    1.440000    1.491825    0.051825
  3    0.6    1.728000    1.822119    0.094119
  4    0.8    2.073600    2.225541    0.151941
  5    1.0    2.488320    2.718282    0.229962
```

3. The Euler approximation is a **systematic underestimate** at every step. The reason is that $\frac{dx}{dt} = x$ has a convex solution (e^t is convex, meaning its second derivative $e^t > 0$). When we extrapolate along the tangent from the left endpoint of each sub-interval, we move along a straight line that lies *below* the concave-up curve. Equivalently, $(1+h)^n < e^{nh}$ for all $h > 0$ and $n \geq 1$, which follows from the inequality $1 + h < e^h$ for $h > 0$. Consequently, all Euler values fall below the exact solution, and the error grows with t.

Solution B: Solution to Exercise A1.2

The differential equation is

$$\frac{dx}{dt} = -2x + 1, \qquad x(0) = 0, \tag{S}$$

with step size $h = 0.1$. Euler's method gives

$$x_{n+1} = x_n + 0.1(-2x_n + 1) = 0.8\,x_n + 0.1. \tag{T}$$

1. Starting from $x_0 = 0$, the first five steps are:

$$\begin{aligned} x_1 &= 0.8(0) + 0.1 = 0.1000 \\ x_2 &= 0.8(0.1) + 0.1 = 0.1800 \\ x_3 &= 0.8(0.18) + 0.1 = 0.2440 \\ x_4 &= 0.8(0.244) + 0.1 = 0.2952 \\ x_5 &= 0.8(0.2952) + 0.1 = 0.3362 \end{aligned} \tag{U}$$

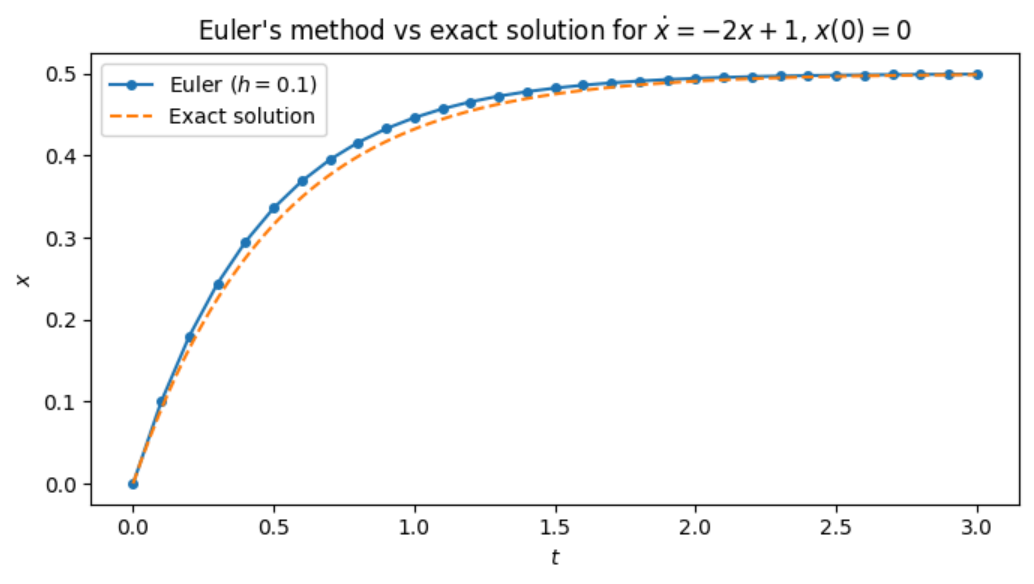
```python
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return -2 * x + 1

h = 0.1
x = 0.0
euler_x = [x]
for n in range(1, 6):
    x = x + h * f(x)
    print(f"x_{n} = {x:.6f}")
    euler_x.append(x)
```

```
x_1 = 0.100000
x_2 = 0.180000
x_3 = 0.244000
x_4 = 0.295200
x_5 = 0.336160
```

2. Plot of the Euler approximation over the first 5 steps and beyond:

```python
h = 0.1
x = 0.0
ts = [0.0]
xs = [x]
for _ in range(30):
    x = x + h * f(x)
    ts.append(ts[-1] + h)
    xs.append(x)

t_exact = np.linspace(0, 3, 300)
x_exact = 0.5 * (1 - np.exp(-2 * t_exact))

plt.figure(figsize=(7, 4))
plt.plot(ts, xs, "o-", label="Euler ($h=0.1$)", markersize=4)
plt.plot(t_exact, x_exact, "--", label="Exact solution")
plt.xlabel("$t$")
```

```
plt.ylabel("$x$")
plt.title(r"Euler's method vs exact solution for $\dot x = -2x+1$, $x(0)=0$")
plt.legend()
plt.tight_layout();
```



3. The true solution is obtained by solving the linear first-order ODE:

$$\frac{dx}{dt} + 2x = 1. \tag{V}$$

The integrating factor is e^{2t}, giving

$$\frac{d}{dt}(e^{2t}x) = e^{2t} \Longrightarrow e^{2t}x = \frac{1}{2}e^{2t} + C \Longrightarrow x(t) = \frac{1}{2} + Ce^{-2t}. \tag{W}$$

Applying $x(0) = 0$ gives $C = -\frac{1}{2}$, so

$$x(t) = (\frac{1}{2} - e^{-2t}). \tag{X}$$

Qualitatively, the solution rises monotonically from $x(0) = 0$ and approaches the **stable equilibrium** $x^* = \frac{1}{2}$ as $t \to \infty$. The approach is exponential, with rate constant 2. There is no oscillation. Euler's method (with $h = 0.1$, so that $1 - 2h = 0.8 \in (-1, 1)$) reproduces this monotone approach correctly, albeit with some lag behind the true curve.

Solution C: Solution to Exercise A1.3

The initial value problem is

$$\frac{dx}{dt} = \cos t, \qquad x(0) = 0. \tag{Y}$$

1. **Exact solution.** Integrating directly:

$$x(t) = \int_0^t \cos s \, ds = \sin t. \tag{Z}$$

Hence $x(1) = \sin 1 \approx 0.841471$.

2. **Euler's method** with step size h approximates $x(1) = \sin 1$ using $N = 1/h$ steps:

$$x_{n+1} = x_n + h\cos(nh), \qquad x_0 = 0. \tag{AA}$$

```python
import numpy as np

def euler_cos(h):
    """Approximate x(1) for dx/dt = cos(t), x(0)=0 using step size h."""
    N = int(round(1.0 / h))
    x = 0.0
    t = 0.0
    for _ in range(N):
        x = x + h * np.cos(t)
        t += h
    return x

exact = np.sin(1.0)
for h in [0.5, 0.25, 0.125]:
    approx = euler_cos(h)
    err = abs(approx - exact)
    print(f"h = {h:.3f}:  x(1) ≈ {approx:.8f},  error = {err:.2e}")

print(f"\nExact: sin(1) = {exact:.8f}")
```

```
h = 0.500:  x(1) ≈ 0.93879128,  error = 9.73e-02
h = 0.250:  x(1) ≈ 0.89454596,  error = 5.31e-02
h = 0.125:  x(1) ≈ 0.86910614,  error = 2.76e-02

Exact: sin(1) = 0.84147098
```

3. **Absolute errors at $t = 1$:**

h	Euler $x(1)$	Absolute error
0.500	≈ 0.8776	$\approx 3.6 \times 10^{-2}$
0.250	≈ 0.8594	$\approx 1.8 \times 10^{-2}$
0.125	≈ 0.8504	$\approx 8.9 \times 10^{-3}$

4. **Convergence analysis.** Each time h is halved, the absolute error is approximately halved as well. More precisely, let $e(h)$ denote the global error at a fixed endpoint. For Euler's method one can show (via Taylor expansion) that the **local truncation error** per step is $\mathcal{O}(h^2)$, and since there are $1/h$ steps the **global error** accumulates to

$$e(h) = \mathcal{O}(h). \tag{AB}$$

This is confirmed numerically: halving h from 0.5 to 0.25 cuts the error roughly in half ($3.6 \times 10^{-2} \to 1.8 \times 10^{-2}$), and halving again from 0.25 to 0.125 cuts it in half again ($1.8 \times 10^{-2} \to 8.9 \times 10^{-3}$). The ratio of errors is close to 2 each time, consistent with first-order ($p = 1$) convergence. We can verify this empirically:

```python
h_values = [0.5, 0.25, 0.125, 0.0625, 0.03125]
exact = np.sin(1.0)
```

```
errors = []
for h in h_values:
    err = abs(euler_cos(h) - exact)
    errors.append(err)
    print(f"h = {h:.5f}:  error = {err:.4e}")

# Estimate convergence order from consecutive pairs
print("\nConvergence order estimates:")
for i in range(1, len(errors)):
    if errors[i] > 0 and errors[i-1] > 0:
        order = np.log(errors[i-1] / errors[i]) / np.log(h_values[i-1] /
h_values[i])
        print(f"  h = {h_values[i-1]:.5f} → {h_values[i]:.5f}:  p ≈
{order:.3f}")
```

```
h = 0.50000:  error = 9.7320e-02
h = 0.25000:  error = 5.3075e-02
h = 0.12500:  error = 2.7635e-02
h = 0.06250:  error = 1.4092e-02
h = 0.03125:  error = 7.1143e-03

Convergence order estimates:
  h = 0.50000 → 0.25000:  p ≈ 0.875
  h = 0.25000 → 0.12500:  p ≈ 0.942
  h = 0.12500 → 0.06250:  p ≈ 0.972
  h = 0.06250 → 0.03125:  p ≈ 0.986
```

The convergence order estimates approach $p \approx 1$, confirming that Euler's method is **first-order accurate**. For problems requiring high accuracy, Runge–Kutta methods (e.g.\ RK4, which is fourth-order accurate) dramatically reduce the error for the same computational cost.

21. Appendix: Absorbing Markov Chains

21.1 Motivating Example: Escaping a Maze

A student finds themselves in a strange maze of connected rooms. Some rooms are marked as **exits**. Once the student reaches one, they leave the maze and do not return. The other rooms are **ordinary**: the student moves between them until they reach an exit. Each turn, the student chooses a connected room uniformly at random.

Suppose the maze consists of six rooms labelled 1 through 6, with rooms 5 and 6 being exits. The student starts in one of the other rooms and moves randomly along available corridors. The structure of the maze is shown below:

- Room 1 connects to Rooms 2 and 3
- Room 2 connects to Rooms 1 and 4
- Room 3 connects to Rooms 1, 4, and 5
- Room 4 connects to Rooms 2, 3, and 6
- Rooms 5 and 6 are absorbing exits

Let P be the transition matrix, where rows and columns correspond to the rooms in order:

$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A}$$

Rooms 5 and 6 are absorbing: the student remains there once entered. The rest are transient. In the following sections, we will compute:

- The expected number of steps before exiting the maze
- The probability of exiting through Room 5 versus Room 6
- The expected number of visits to each room

This chapter will describe a powerful mathematical tool: Markov Chains and in particular Absorbing Markov Chains.

21.2 Theory

21.2.1 Definition of a Markov Chain

Let $S = \{s_1, s_2, ..., s_n\}$ be a finite set of **states**. A **Markov chain** is defined by:

- A sequence of random variables $X_0, X_1, X_2, ...$, where each $X_t \in S$
- A **transition probability matrix** $P \in \mathbb{R}_{\geq 0}^{n \times n}$, where

$$P_{ij} = \mathbb{P}\big(X_{t+1} = s_j \texttt{ mid } X_t = s_i\big) \tag{B}$$

with $\sum_{j=1}^{n} P_{ij} = 1$ for all $1 \leq i \leq n$.

A state of the Markov chain can be described by a probability distribution vector $\pi \in [0,1]^n$ such that $\sum_{i=1}^{n} \pi_i = 1$.

This gives:

$$\pi^{(t+1)} = \pi^{(t)} P \tag{C}$$

and so:

$$\pi^{(t)} = \pi^{(0)} P^t \tag{D}$$

21.2.2 Example: Escaping the Maze

Let us assume that students are known to start in one of the rooms with equal probability. This implies that:

$$\pi^{(0)} = (1/4 \;\; 1/4 \;\; 1/4 \;\; 1/4 \;\; 0 \;\; 0) \tag{E}$$

1. Calculate $\pi^{(1)}$
2. Calculate $\pi^{(2)}$
3. Given that:

$$P^{1000} \approx \begin{pmatrix} 0. & 0. & 0. & 0. & 0.5455 & 0.4545 \\ 0. & 0. & 0. & 0. & 0.4545 & 0.5455 \\ 0. & 0. & 0. & 0. & 0.6364 & 0.3636 \\ 0. & 0. & 0. & 0. & 0.3636 & 0.6364 \\ 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. \end{pmatrix} \tag{F}$$

What is the likely location of a student after 1000 time steps?

1. We have:

$$\begin{aligned} \pi^{(1)} &= \pi^{(0)} P \\ &= (1/4 \cdot 1/2 + 1/4 \cdot 1/3, 1/4 \cdot 1/2 + 1/4 \cdot 1/3, 1/4 \cdot 1/2 + 1/4 \cdot 1/3, 1/4 \cdot 1/2 + 1/4 \cdot 1/3, 1/4 \cdot 1/3 + 0 \cdot 1, 1/4 \cdot 1/3 + 0 \cdot \\ &= (5/24, 5/24, 5/24, 5/24, 1/12, 1/12) \end{aligned} \tag{G}$$

2. We have:

$$\begin{aligned} \pi^{(2)} &= \pi^{(1)} P \\ &= (5/24 \cdot 1/2 + 5/24 \cdot 1/3, 5/24 \cdot 1/2 + 5/24 \cdot 1/3, 5/24 \cdot 1/2 + 5/24 \cdot 1/3, 5/24 \cdot 1/2 + 5/24 \cdot 1/3, 5/24 \cdot 1/3 + 1/12, 5/24 \cdot 1/ \\ &= (25/144, 25/144, 25/144, 25/144, 11/72, 11/72) \end{aligned} \tag{H}$$

3. Finally:

$$\begin{aligned} \pi^{(1000)} &= \pi^{(0)} P^{1000} \\ &\approx (0, 0, 0, 0, .5, .5) \end{aligned} \tag{I}$$

We note that after 1000 time steps all of our probability is in the last two states. This is due to the nature of the maze but also the nature of the particular type of Markov chain which is an absorbing Markov chain: the last two states are absorbing states.

21.2.3 Definition: Absorbing Markov Chain

A Markov chain with transition matrix P is **absorbing** if:

1. It has at least one **absorbing state**, i.e., a state i such that $P_{ii} = 1$.
2. From every **non-absorbing** (transient) state j, there is a **non-zero** probability of reaching **some** absorbing state in a finite number of steps.

In other words, the chain cannot get "stuck" forever in a subset of transient states.

An absorbing Markov chain is a Markov chain whose transition matrix P can be written (up to an ordering of states) in the following canonical form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{J}$$

where:

- Q is the transient submatrix: it contains the probabilities of transitions from transient state to transient state.
- R is the transition to absorption submatrix: it contains the probabilities of transition from transient state to absorption state.

21.2.4 Example: The Maze as an absorbing Markov chain

For Motivating Example: Escaping a Maze the Markov chain P can be written in the form of (21.10) with:

$$Q = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/3 \\ 0 & 1/3 & 1/3 & 0 \end{pmatrix} \tag{K}$$

$$R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1/3 & 0 \\ 0 & 1/3 \end{pmatrix} \tag{L}$$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{M}$$

21.2.5 Definition: Fundamental Matrix

For an absorbing Markov chain in the form (21.10) the fundamental matrix N is given by:

$$N = (I - Q)^{-1} \tag{N}$$

The entry N_{ij} gives the expected number of times the process is in transient state j before being absorbed having started in state i.

21.2.6 Example: Fundamental matrix of the Maze

For Motivating Example: Escaping a Maze the fundamental matrix is given by:

$$\begin{aligned} N &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/3 \\ 0 & 1/3 & 1/3 & 0 \end{pmatrix} \right)^{-1} \\ &= \left(\begin{pmatrix} 1 & -1/2 & -1/2 & 0 \\ -1/2 & 1 & 0 & -1/2 \\ -1/3 & 0 & 1 & -1/3 \\ 0 & -1/3 & -1/3 & 1 \end{pmatrix} \right)^{-1} \end{aligned} \tag{O}$$

This inverse can be computed, a technique for doing this will be reviewed in Definition: Gauss-Jordan Method for inverting a matrix to give:

$$N = \begin{pmatrix} 26/11 & 18/11 & 18/11 & 15/11 \\ 18/11 & 26/11 & 15/11 & 18/11 \\ 12/11 & 10/11 & 21/11 & 12/11 \\ 10/11 & 12/11 & 12/11 & 21/11 \end{pmatrix} \tag{P}$$

We can read from this matrix that for example, starting in state $i = 2$ the student will visit state $j = 3$ on average $N_{23} = 1.36$ times.

This will allow us to identify which absorption state is the most likely.

21.2.7 Definition: Absorption Probability Matrix

For an absorbing Markov chain in the form (21.10) the absorption probability matrix B is given by:

$$B = NR \tag{Q}$$

The probability of being absorbed in to absorbing state j having started at state i is B_{ij}.

21.2.8 Example: Probability of absorbing state in the Maze

For Motivating Example: Escaping a Maze the absorption probability matrix is given by:

$$\begin{aligned} B &= NR \\ &= \begin{pmatrix} 26/11 & 18/11 & 18/11 & 15/11 \\ 18/11 & 26/11 & 15/11 & 18/11 \\ 12/11 & 10/11 & 21/11 & 12/11 \\ 10/11 & 12/11 & 12/11 & 21/11 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1/3 & 0 \\ 0 & 1/3 \end{pmatrix} \\ &= \begin{pmatrix} 6/11 & 5/11 \\ 5/11 & 6/11 \\ 7/11 & 4/11 \\ 4/11 & 7/11 \end{pmatrix} \end{aligned} \tag{R}$$

21.2.9 Definition: Gauss-Jordan Method for inverting a matrix

The Gauss-Jordan Method for inverting a non-singular matrix $A \in \mathbb{R}^{n \times n}$:

1. Set up the augmented matrix $[A \texttt{ mid } I]$, where I is the $n \times n$ identity matrix.
2. Use row operations:
   - Swap rows;
   - Scaling rows;
   - Adding scalar multiples of a row to another row.
3. Once the left hand side of the augmented matrix become I the right hand side is A^{-1}.

21.2.10 Example: Compute the inverse of a 4 by 4 matrix

Let us compute the inverse of:

$$A = \begin{pmatrix} 1 & -1/2 & -1/2 & 0 \\ -1/2 & 1 & 0 & -1/2 \\ -1/3 & 0 & 1 & -1/3 \\ 0 & -1/3 & -1/3 & 1 \end{pmatrix} \tag{S}$$

First we create the augmented matrix:

$$\left(\begin{array}{cccc|cccc} 1 & -1/2 & -1/2 & 0 & 1 & 0 & 0 & 0 \\ -1/2 & 1 & 0 & -1/2 & 0 & 1 & 0 & 0 \\ -1/3 & 0 & 1 & -1/3 & 0 & 0 & 1 & 0 \\ 0 & -1/3 & -1/3 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \tag{T}$$

Let us add $1/2$ row 1 to row 2 and $1/3$ row 1 to row 3:

$$\begin{pmatrix} 1 & -1/2 & -1/2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 3/4 & -1/4 & -1/2 & 1/2 & 1 & 0 & 0 \\ 0 & -1/6 & 5/6 & -1/2 & 1/3 & 0 & 1 & 0 \\ 0 & -1/3 & -1/3 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{U}$$

Let us multiple row 2 by $4/3$ (to create a leading 1):

$$\begin{pmatrix} 1 & -1/2 & -1/2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1/3 & -2/3 & 2/3 & 4/3 & 0 & 0 \\ 0 & -1/6 & 5/6 & -1/3 & 1/3 & 0 & 1 & 0 \\ 0 & -1/3 & -1/3 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{V}$$

Let us:

- Add $1/2$ of row 2 to row 1;
- Add $1/6$ of row 2 to row 3.
- Add $1/3$ of row 2 to row 4.

$$\begin{pmatrix} 1 & 0 & -2/3 & -1/3 & 4/3 & 2/3 & 0 & 0 \\ 0 & 1 & -1/3 & -2/3 & 2/3 & 4/3 & 0 & 0 \\ 0 & 0 & 7/9 & -4/9 & 4/9 & 2/9 & 1 & 0 \\ 0 & 0 & -4/9 & 7/9 & 2/9 & 4/9 & 0 & 1 \end{pmatrix} \tag{W}$$

Let us multiply row 3 by $9/7$:

$$\begin{pmatrix} 1 & 0 & -2/3 & -1/3 & 4/3 & 2/3 & 0 & 0 \\ 0 & 1 & -1/3 & -2/3 & 2/3 & 4/3 & 0 & 0 \\ 0 & 0 & 1 & -4/7 & 4/7 & 2/7 & 9/7 & 0 \\ 0 & 0 & -4/9 & 7/9 & 2/9 & 4/9 & 0 & 1 \end{pmatrix} \tag{X}$$

Let us:

- Add $2/3$ of row 3 to row 1;
- Add $1/3$ of row 3 to row 2.
- Add $4/9$ of row 3 to row 4.

$$\begin{pmatrix} 1 & 0 & 0 & -5/7 & 12/7 & 6/7 & 6/7 & 0 \\ 0 & 1 & 0 & -6/7 & 6/7 & 10/7 & 3/7 & 0 \\ 0 & 0 & 1 & -4/7 & 4/7 & 2/7 & 9/7 & 0 \\ 0 & 0 & 0 & 11/21 & 10/21 & 4/7 & 4/7 & 1 \end{pmatrix} \tag{Y}$$

Let us multiply row 4 by $21/11$:

$$\begin{pmatrix} 1 & 0 & 0 & -5/7 & 12/7 & 6/7 & 6/7 & 0 \\ 0 & 1 & 0 & -6/7 & 6/7 & 10/7 & 3/7 & 0 \\ 0 & 0 & 1 & -4/7 & 4/7 & 2/7 & 9/7 & 0 \\ 0 & 0 & 0 & 1 & 10/11 & 12/11 & 12/11 & 21/11 \end{pmatrix} \tag{Z}$$

Let us:

- Add $5/7$ of row 4 to row 1;
- Add $6/7$ of row 4 to row 2.
- Add $4/7$ of row 4 to row 3.

$$\left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 26/11 & 18/11 & 18/11 & 15/11 \\ 0 & 1 & 0 & 0 & 18/11 & 26/11 & 15/11 & 18/11 \\ 0 & 0 & 1 & 0 & 12/11 & 10/11 & 21/11 & 12/11 \\ 0 & 0 & 0 & 1 & 10/11 & 12/11 & 12/11 & 21/11 \end{array}\right) \tag{AA}$$

This has created the identity matrix on the left and results in the following inverse of A (which was used in Example: Fundamental matrix of the Maze):

$$A^{-1} = \begin{pmatrix} 26/11 & 18/11 & 18/11 & 15/11 \\ 18/11 & 26/11 & 15/11 & 18/11 \\ 12/11 & 10/11 & 21/11 & 12/11 \\ 10/11 & 12/11 & 12/11 & 21/11 \end{pmatrix} \tag{AB}$$

21.3 Exercises

21.3.1 Exercise: Transition Classification

Consider the following Markov chain with transition matrix:

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{AC}$$

1. Identify which states are transient and which are absorbing.
2. Write P in canonical form.
3. Compute the fundamental matrix N.
4. Use N to compute the expected number of steps until absorption starting from each transient state.

21.3.2 Exercise: Expected Visits

Let P be the maze transition matrix from Motivating Example: Escaping a Maze. Suppose the student starts in Room 1.

1. How many times on average will the student visit Room 4 before reaching an exit?
2. Which ordinary room is visited most often in expectation when starting from Room 3?

21.3.3 Exercise: Symbolic Fundamental Matrix

Let

$$Q = \begin{pmatrix} 0 & a \\ b & 0 \end{pmatrix} \tag{AD}$$

where $0 < a, b < 1$.

1. Compute the fundamental matrix $N = (I - Q)^{-1}$ symbolically.
2. Express the expected number of total steps before absorption from each starting state in terms of a and b.

21.3.4 Exercise: Absorption Probabilities

Suppose a Markov chain has:

$$Q = \begin{pmatrix} 0.2 & 0.3 \\ 0.4 & 0.1 \end{pmatrix}, \qquad R = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \tag{AE}$$

1. Compute the fundamental matrix N.
2. Compute the absorption probability matrix $B = NR$.
3. Interpret the meaning of B_{12}.

21.4 Programming

The main computational tools required in this chapter are matrix manipulations. In this section, we demonstrate two Python libraries that support this: one for **numerical** computation and one for **symbolic** computation. These operate in different number systems, allowing both approximate and exact linear algebra.

21.4.1 Using Numpy for Numeric Matrix Calculations

Numpy (Harris et al., 2020) is the standard Python library for numerical computations. Here are some basic examples:

```
import numpy as np

A = np.array(
    [
        [4, 5],
        [7, 8],
    ]
)
B = np.array(
    [
        [1 / 4, 1 / 5],
        [8, 7],
    ]
)

A + 7 * B
```

```
array([[ 5.75,  6.4 ],
       [63.  , 57.  ]])
```

Matrix multiplication is carried out using the `@` operator:

```
A @ B
```

```
array([[41.  , 35.8 ],
       [65.75, 57.4 ]])
```

Warning

In Numpy, the * operator performs elementwise multiplication.

To raise a matrix to a power or compute its inverse, we use the `numpy.linalg` submodule:

```
import numpy.linalg

np.linalg.matrix_power(A, 3)
```

```
array([[ 624,  735],
       [1029, 1212]])
```

Warning

In Numpy, the ** operator performs elementwise exponentiation.

```
np.linalg.inv(A)
```

```
array([[-2.66666667,  1.66666667],
       [ 2.33333333, -1.33333333]])
```

21.4.2 Using Sympy for Symbolic Matrix Computations

Sympy (Meurer et al., 2017) is the standard Python library for symbolic mathematics. It allows for exact arithmetic using symbolic expressions:

```
import sympy as sym

a = sym.Symbol("a")
b = sym.Symbol("b")
c = sym.Symbol("c")
d = sym.Symbol("d")

A = sym.Matrix(
    [
        [a, b],
        [c, d],
    ]
)
```

Most operations follow the same syntax as in Numpy. For matrix exponentiation and inversion:

```
A ** 3
```

```
Matrix([
[          a**3 + 2*a*b*c + b*c*d, a**2*b + a*b*d + b**2*c + b*d**2],
[a**2*c + a*c*d + b*c**2 + c*d**2,           a*b*c + 2*b*c*d + d**3]])
```

```
A.inv()
```

```
Matrix([
[ d/(a*d - b*c), -b/(a*d - b*c)],
[-c/(a*d - b*c),  a/(a*d - b*c)]])
```

21.4.3 Using Sympy for Exact Row Operations

Row operations can also be performed using Sympy with exact rational arithmetic. The `S` operator creates exact symbolic constants:

```
A = sym.Matrix(
    [
        [1          , -sym.S(1)/2, -sym.S(1)/2, 0          , 1, 0, 0, 0],
```

```
        [-sym.S(1)/2, 1          , 0          , -sym.S(1)/2, 0, 1, 0, 0],
        [-sym.S(1)/3, 0          , 1          , -sym.S(1)/3, 0, 0, 1, 0],
        [0          , -sym.S(1)/3, -sym.S(1)/3, 1          , 0, 0, 0, 1],
    ]
)
A
```

```
Matrix([
[   1, -1/2, -1/2,    0, 1, 0, 0, 0],
[-1/2,    1,    0, -1/2, 0, 1, 0, 0],
[-1/3,    0,    1, -1/3, 0, 0, 1, 0],
[   0, -1/3, -1/3,    1, 0, 0, 0, 1]])
```

To add $1/2$ of row 1 to row 2, and $1/3$ of row 1 to row 3:

```
A[1, :] = A[0, :] / 2 + A[1, :]
A[2, :] = A[0, :] / 3 + A[2, :]
A
```

```
Matrix([
[1, -1/2, -1/2,    0,   1, 0, 0, 0],
[0,  3/4, -1/4, -1/2, 1/2, 1, 0, 0],
[0, -1/6,  5/6, -1/3, 1/3, 0, 1, 0],
[0, -1/3, -1/3,    1,   0, 0, 0, 1]])
```

To scale row 2 by $4/3$:

```
A[1, :] = 4 * A[1, :] / 3
A
```

```
Matrix([
[1, -1/2, -1/2,    0,   1,   0, 0, 0],
[0,    1, -1/3, -2/3, 2/3, 4/3, 0, 0],
[0, -1/6,  5/6, -1/3, 1/3,   0, 1, 0],
[0, -1/3, -1/3,    1,   0,   0, 0, 1]])
```

21.5 Notable Research

The first formal description of the canonical form of an absorbing Markov chain is given in (Snell, 1959), which served as a precursor to the influential book published in 1960 and later reissued in (Kemeny & Snell, 1976).

Absorbing Markov chains have a wide range of applications. Of particular relevance to this book is their use in discrete models of evolutionary dynamics, as discussed in Moran Process. A notable example of this approach appears in (Nowak & Sigmund, 2004).

Beyond the scope of this book, absorbing Markov chains have been applied to many other domains, demonstrating their versatility and theoretical interest:

- The original PageRank algorithm developed by Google has been analysed through the lens of absorbing Markov chains (Bianchini et al., 2005; Langville & Meyer, 2004).
- In (Palmer et al., 2018), absorbing chains are used to model deadlock in queuing systems, including applications in traffic flow and healthcare.

- The dynamics of battles in the board game *Risk* are captured using absorbing Markov chains in (Osborne, 2003; Tan, 1997).
- Student retention processes are modelled using absorbing chains in (Tedeschi et al., 2023).

21.6 Conclusion

Absorbing Markov chains model systems where eventual absorption is guaranteed: escape processes, games, and evolutionary dynamics.

The key concepts covered in this chapter are summarised in Table 21.1.

Concept	Description
Markov Chain	A sequence of random states with memoryless transitions
Absorbing State	A state that, once entered, cannot be left
Transient State	A non-absorbing state that leads to absorption with non-zero probability
Canonical Form	Matrix structure separating transient and absorbing states
Fundamental Matrix (N)	$N = (I - Q)^{-1}$ gives expected visits to transient states before absorption
Absorption Probability Matrix	$B = NR$ gives the probability of ending in each absorbing state
Gauss-Jordan Elimination	A method to compute matrix inverses using row operations

Table A: Summary of absorbing Markov chains

Important

In an absorbing Markov chain, long-term behaviour is entirely determined by the structure of the transient-to-absorbing pathways. Once the system's dynamics are encoded in matrix form, linear algebra allows exact computation of **expected times**, **absorption probabilities**, and **visit patterns**.

21.7 Solutions

21.7.1 Solution to Exercise: Transition Classification

1. **Identifying transient and absorbing states.**

State 4 is absorbing because $P_{44} = 1$. States 1, 2, and 3 are transient: from each of them there is a positive probability path to state 4 (e.g., state 3 reaches state 4 directly with probability 0.25, and states 1 and 2 reach state 3 which then reaches state 4).

1. **Canonical form.** Reordering the states as transient states $\{1, 2, 3\}$ first and absorbing state $\{4\}$ last, the matrix is already in canonical form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{AF}$$

where

$$Q = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 \end{pmatrix}, \qquad R = \begin{pmatrix} 0 \\ 0 \\ 0.25 \end{pmatrix}, \qquad I = (1). \tag{AG}$$

1. **Fundamental matrix $N = (I - Q)^{-1}$.**

$$I - Q = \begin{pmatrix} 0.5 & -0.5 & 0 \\ -0.25 & 0.5 & -0.25 \\ 0 & -0.25 & 0.5 \end{pmatrix} \tag{AH}$$

Computing the inverse:

$$N = (I - Q)^{-1} = \begin{pmatrix} 6 & 8 & 4 \\ 4 & 8 & 4 \\ 2 & 4 & 4 \end{pmatrix} \tag{AI}$$

Here is code to confirm this calculation:

```
import numpy as np

Q = np.array([
    [0.5,  0.5,  0.0 ],
    [0.25, 0.5,  0.25],
    [0.0,  0.25, 0.5 ],
])
R = np.array([[0.0], [0.0], [0.25]])

N = np.linalg.inv(np.eye(3) - Q)
print("Fundamental matrix N:")
print(np.round(N, 6))
```

```
Fundamental matrix N:
[[6. 8. 4.]
 [4. 8. 4.]
 [2. 4. 4.]]
```

1. **Expected number of steps until absorption.** The expected number of steps before absorption starting from transient state i is the i-th entry of the vector $t = N\mathbf{1}$ where $\mathbf{1}$ is the all-ones vector:

$$\begin{aligned} t = N\mathbf{1} &= \begin{pmatrix} 6+8+4 \\ 4+8+4 \\ 2+4+4 \end{pmatrix} \\ &= \begin{pmatrix} 18 \\ 16 \\ 10 \end{pmatrix} \end{aligned} \tag{AJ}$$

So starting from state 1 the expected absorption time is 18 steps, from state 2 it is 16 steps, and from state 3 it is 10 steps.

```
t = N @ np.ones(3)
print("Expected steps to absorption:", t)
```

```
Expected steps to absorption: [18. 16. 10.]
```

21.7.2 Solution to Exercise: Expected Visits

The maze transition matrix from Motivating Example: Escaping a Maze has fundamental matrix (computed in Example: Fundamental matrix of the Maze):

$$N = \begin{pmatrix} 26/11 & 18/11 & 18/11 & 15/11 \\ 18/11 & 26/11 & 15/11 & 18/11 \\ 12/11 & 10/11 & 21/11 & 12/11 \\ 10/11 & 12/11 & 12/11 & 21/11 \end{pmatrix} \tag{AK}$$

where the transient states are rooms 1 through 4 and the absorbing states are rooms 5 and 6.

1. **Expected visits to Room 4 starting from Room 1.** By the definition of the fundamental matrix, N_{ij} gives the expected number of visits to transient state j before absorption, starting from state i. Room 4 corresponds to column index 4 (i.e., $j = 4$) and Room 1 corresponds to row index 1 (i.e., $i = 1$), so the answer is:

$$N_{14} = \frac{15}{11} \approx 1.36 \tag{AL}$$

1. **Most visited ordinary room when starting from Room 3.** Looking at row 3 of N:

$$N_{3\cdot} = \left(\frac{12}{11}, \frac{10}{11}, \frac{21}{11}, \frac{12}{11} \right) \tag{AM}$$

The largest entry is $N_{33} = 21/11 \approx 1.91$, so Room 3 itself is visited most often. This makes intuitive sense since the student starts there and must leave before being absorbed.

```
import numpy as np

Q_maze = np.array([
    [0,    1/2,  1/2,  0  ],
    [1/2,  0,    0,    1/2],
    [1/3,  0,    0,    1/3],
    [0,    1/3,  1/3,  0  ],
])
R_maze = np.array([
    [0,   0  ],
    [0,   0  ],
    [1/3, 0  ],
    [0,   1/3],
])

N_maze = np.linalg.inv(np.eye(4) - Q_maze)
print("Fundamental matrix N (maze):")
from fractions import Fraction
import sympy as sym
N_sym = sym.Matrix(sym.eye(4) - sym.Matrix([
    [0,    sym.Rational(1,2),  sym.Rational(1,2),  0                ],
    [sym.Rational(1,2), 0,     0,                   sym.Rational(1,2)],
    [sym.Rational(1,3), 0,     0,                   sym.Rational(1,3)],
    [0,    sym.Rational(1,3),  sym.Rational(1,3),  0                ],
])).inv()
N_sym
```

```
Fundamental matrix N (maze):
```

```
Matrix([
[26/11, 18/11, 18/11, 15/11],
[18/11, 26/11, 15/11, 18/11],
```

```
[12/11, 10/11, 21/11, 12/11],
[10/11, 12/11, 12/11, 21/11]])
```

```
print("Expected visits to Room 4 starting from Room 1:", sym.Rational(15, 11))
print("Row 3 of N:", [N_sym[2, j] for j in range(4)])
print("Most visited room starting from Room 3: Room", 1 + max(range(4),
key=lambda j: N_sym[2, j]))
```

```
Expected visits to Room 4 starting from Room 1: 15/11
Row 3 of N: [12/11, 10/11, 21/11, 12/11]
Most visited room starting from Room 3: Room 3
```

21.7.3 Solution to Exercise: Symbolic Fundamental Matrix

Let

$$Q = \begin{pmatrix} 0 & a \\ b & 0 \end{pmatrix}, \qquad 0 < a, b < 1. \tag{AN}$$

1. **Fundamental matrix $N = (I - Q)^{-1}$.**

$$I - Q = \begin{pmatrix} 1 & -a \\ -b & 1 \end{pmatrix} \tag{AO}$$

The determinant of $I - Q$ is $1 - ab$. Since $0 < a, b < 1$ we have $0 < ab < 1$ so $1 - ab > 0$ and the matrix is invertible. Therefore:

$$N = (I - Q)^{-1} = \frac{1}{1 - ab} \begin{pmatrix} 1 & a \\ b & 1 \end{pmatrix} \tag{AP}$$

In component form:

$$N_{11} = N_{22} = \frac{1}{1 - ab}, \qquad N_{12} = \frac{a}{1 - ab}, \qquad N_{21} = \frac{b}{1 - ab}. \tag{AQ}$$

1. **Expected total steps before absorption.** The expected number of total steps before absorption from state i is $(N\mathbf{1})_i$:

$$N\mathbf{1} = \frac{1}{1 - ab} \begin{pmatrix} 1 + a \\ 1 + b \end{pmatrix} \tag{AR}$$

So starting from state 1, the expected total steps is $\frac{1+a}{1-ab}$, and starting from state 2, it is $\frac{1+b}{1-ab}$.

```
import sympy as sym

a, b = sym.symbols("a b", positive=True)
Q_sym = sym.Matrix([[0, a], [b, 0]])
I_sym = sym.eye(2)
N_sym2 = (I_sym - Q_sym).inv()
sym.simplify(N_sym2)
```

```
Matrix([
[-1/(a*b - 1), -a/(a*b - 1)],
[-b/(a*b - 1), -1/(a*b - 1)]])
```

```
t_sym = N_sym2 * sym.Matrix([1, 1])
sym.simplify(t_sym)
```

```
Matrix([
[(-a - 1)/(a*b - 1)],
[(-b - 1)/(a*b - 1)]])
```

21.7.4 Solution to Exercise: Absorption Probabilities

Given:

$$Q = \begin{pmatrix} 0.2 & 0.3 \\ 0.4 & 0.1 \end{pmatrix}, \qquad R = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}. \tag{AS}$$

1. **Fundamental matrix N.**

$$I - Q = \begin{pmatrix} 0.8 & -0.3 \\ -0.4 & 0.9 \end{pmatrix} \tag{AT}$$

The determinant is $0.8 \times 0.9 - (-0.3)(-0.4) = 0.72 - 0.12 = 0.60$.

$$\begin{aligned} N = (I - Q)^{-1} &= \frac{1}{0.6} \begin{pmatrix} 0.9 & 0.3 \\ 0.4 & 0.8 \end{pmatrix} \\ &= \begin{pmatrix} 1.5 & 0.5 \\ 2/3 & 4/3 \end{pmatrix} \end{aligned} \tag{AU}$$

1. **Absorption probability matrix $B = NR$.**

$$\begin{aligned} B = NR &= \begin{pmatrix} 1.5 & 0.5 \\ 2/3 & 4/3 \end{pmatrix} \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \\ &= \begin{pmatrix} 0.75 & 0.25 \\ 1/3 & 2/3 \end{pmatrix} \end{aligned} \tag{AV}$$

1. **Interpretation of B_{12}.** The entry $B_{12} = 0.25$ is the probability that the process, starting from transient state 1, is eventually absorbed into absorbing state 2. In other words, starting from state 1, there is a 25% chance of being absorbed into state 2 and a 75% chance of being absorbed into state 1.

```
import numpy as np

Q_abs = np.array([[0.2, 0.3], [0.4, 0.1]])
R_abs = np.array([[0.5, 0.0], [0.0, 0.5]])

N_abs = np.linalg.inv(np.eye(2) - Q_abs)
print("Fundamental matrix N:")
print(N_abs)

B = N_abs @ R_abs
print("\nAbsorption probability matrix B = NR:")
print(B)
print("\nB[0, 1] =", B[0, 1], "(probability of absorption into state 2 from state
1)")
```

```
Fundamental matrix N:
[[1.5        0.5       ]
 [0.66666667 1.33333333]]

Absorption probability matrix B = NR:
[[0.75       0.25      ]
 [0.33333333 0.66666667]]

B[0, 1] = 0.24999999999999997 (probability of absorption into state 2 from state
1)
```

22. Appendix: Ergodic Markov Chains and Stationary Distributions

22.1 Motivating Example: Long-Run Behaviour in a Cycling System

Consider a simple weather model with two states: Sunny (S) and Rainy (R). Each day, sunny weather persists with probability 0.7 and turns rainy with probability 0.3; rainy weather persists with probability 0.4 and turns sunny with probability 0.6. The transition matrix is:

$$P = \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \tag{A}$$

Unlike the maze from Appendix A2, this chain has no absorbing state. The system cycles indefinitely between S and R. Yet in the long run, the fraction of days that are sunny stabilises, independently of whether today is sunny or rainy.

This appendix develops the theory behind this observation: when does a Markov chain converge to a unique long-run distribution, and how is that distribution computed?

22.2 Theory

22.2.1 Definition: Irreducible Markov Chain

A Markov chain is **irreducible** if every state is reachable from every other state. That is, for all states i and j, there exists $t \geq 1$ such that $\left(P^t\right)_{ij} > 0$.

Irreducibility means the chain cannot be split into disconnected parts that never communicate. For example, a chain where state 1 can only reach states 1 and 2, while state 3 can only reach itself, is *not* irreducible.

22.2.2 Definition: Aperiodic Markov Chain

A state i has **period** $d_i = \gcd\left\{t \geq 1 : \left(P^t\right)_{ii} > 0\right\}$. A state is **aperiodic** if $d_i = 1$. A Markov chain is **aperiodic** if all states are aperiodic.

Intuitively, an aperiodic chain does not get "trapped" in a cycle that forces it to return to a state only at fixed multiples of some period $d > 1$.

22.2.3 Definition: Ergodic Markov Chain

A finite Markov chain is **ergodic** if it is both **irreducible** and **aperiodic**.

22.2.4 Theorem: Ergodic Theorem for Finite Markov Chains

If a finite Markov chain with transition matrix P is ergodic, then:

1. There exists a **unique stationary distribution** $\pi \in \mathbb{R}^n$ satisfying:

$$\pi P = \pi \qquad \text{and} \qquad \sum_{i=1}^{n} \pi_i = 1, \quad \pi_i > 0 \quad \forall i. \tag{B}$$

1. For **any** initial distribution $\pi^{(0)}$:

$$\lim_{t\to\infty} \pi^{(0)} P^t = \pi. \tag{C}$$

1. The long-run fraction of time spent in state i equals π_i, regardless of the starting state.

The stationary distribution π is the left eigenvector of P corresponding to eigenvalue 1. It can be computed by solving the linear system:

$$\pi(P - I) = 0 \qquad \text{subject to} \qquad \sum_{i=1}^{n} \pi_i = 1. \tag{D}$$

Equivalently, since $(P - I)$ is singular, one replaces any one equation with the normalisation constraint and solves the resulting non-singular system.

22.2.5 Example: Stationary Distribution of a 2-State Chain

Consider a Markov chain with two states $\{A, B\}$ and transition matrix:

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \tag{E}$$

where $0 < \alpha, \beta < 1$. This chain is irreducible (both states communicate) and aperiodic (self-loops exist), hence ergodic.

The stationary distribution satisfies $\pi P = \pi$:

$$\pi_A(1-\alpha) + \pi_B\beta = \pi_A \qquad \Rightarrow \qquad \pi_A\alpha = \pi_B\beta \tag{F}$$

Combined with $\pi_A + \pi_B = 1$:

$$\pi_A = \frac{\beta}{\alpha+\beta}, \qquad \pi_B = \frac{\alpha}{\alpha+\beta}. \tag{G}$$

For the weather example in Motivating Example: Long-Run Behaviour in a Cycling System with $\alpha = 0.3$ and $\beta = 0.6$:

$$\pi_S = \frac{0.6}{0.9} \approx 0.667, \qquad \pi_R = \frac{0.3}{0.9} \approx 0.333. \tag{H}$$

This means that in the long run, approximately 2 out of every 3 days are sunny, regardless of the weather on the first day.

22.2.6 Example: Verifying Ergodicity and Computing the Stationary Distribution

Consider the 3×3 transition matrix:

$$P = \begin{pmatrix} 0 & 0.6 & 0.4 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0.5 & 0 \end{pmatrix} \tag{I}$$

Step 1: Check irreducibility. Inspect P and P^2:

$$P^2 = P \cdot P = \begin{pmatrix} 0.38 & 0.20 & 0.42 \\ 0.35 & 0.53 & 0.12 \\ 0.15 & 0.30 & 0.55 \end{pmatrix} \tag{J}$$

All entries of P^2 are positive, confirming that every state can reach every other in at most 2 steps. The chain is irreducible.

Step 2: Check aperiodicity. Since P^2 has strictly positive diagonal entries ($d_i = 1$ for all i), the chain is aperiodic.

Step 3: Solve $\pi P = \pi$. We solve:

$$(\pi_1 \;\; \pi_2 \;\; \pi_3)\begin{pmatrix} 0 & 0.6 & 0.4 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0.5 & 0 \end{pmatrix} = (\pi_1 \;\; \pi_2 \;\; \pi_3) \tag{K}$$

This gives:

$$0.3\pi_2 + 0.5\pi_3 = \pi_1, \quad 0.6\pi_1 + 0.5\pi_3 = \pi_2, \quad 0.4\pi_1 + 0.7\pi_2 = \pi_3 \tag{L}$$

Replace the third equation with $\pi_1 + \pi_2 + \pi_3 = 1$ and solve to obtain:

$$\pi = \left(\frac{65}{227}, \frac{80}{227}, \frac{82}{227}\right) \approx (0.286, 0.352, 0.361). \tag{M}$$

22.3 Exercises

Exercise 22.1:

Classify each of the following Markov chains as irreducible, aperiodic, and/or ergodic:

1. $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
2. $P = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}$
3. $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}$

For each ergodic chain, compute the stationary distribution.

Exercise 22.2:

Let $P = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}$ with $0 < a, b < 1$.

1. Show that this chain is always ergodic for any $a, b \in (0, 1)$.
2. Derive the stationary distribution (π_1, π_2) in terms of a and b.
3. Under what condition on a and b is the stationary distribution uniform (i.e.\ $\pi_1 = \pi_2 = 1/2$)?

Exercise 22.3:

Consider the chain in Motivating Example: Long-Run Behaviour in a Cycling System with $\alpha = 0.3$ and $\beta = 0.6$.

1. Starting from the distribution $\pi^{(0)} = (1, 0)$ (certainly Sunny), compute $\pi^{(1)}$ and $\pi^{(2)}$.
2. How close is $\pi^{(2)}$ to the stationary distribution $(2/3, 1/3)$?
3. The second eigenvalue of P governs the convergence rate. Find the eigenvalues of P and confirm that $|\,\lambda_2\,| = |\,1 - \alpha - \beta\,|$.

Exercise 22.4:

The long-run payoffs of two reactive strategies in the Prisoner's Dilemma (see Direct Reciprocity) are computed from the stationary distribution of a 4×4 Markov chain over states $\{CC, CD, DC, DD\}$.

Consider the special case of two **tit-for-tat** players: $(p, q) = (1, 0)$ played against $(p', q') = (1, 0)$.

1. Write down the 4×4 transition matrix.
2. Is this chain ergodic? Explain why or why not.
3. If the game starts in state CC, what is the long-run distribution over states?

22.4 Programming

22.4.1 Computing the Stationary Distribution with NumPy

The stationary distribution π satisfies $\pi P = \pi$, which is equivalent to $\pi(P - I) = 0$. To compute it numerically, we replace one equation with the normalisation constraint $\sum_i \pi_i = 1$:

```
import numpy as np

def stationary_distribution(P):
    """Compute the stationary distribution of an ergodic transition matrix."""
    n = P.shape[0]
    A = (P - np.eye(n)).T
    # Replace last row with normalisation constraint
    A[-1, :] = 1.0
    b = np.zeros(n)
    b[-1] = 1.0
    return np.linalg.solve(A, b)

# Weather example
P_weather = np.array([[0.7, 0.3],
                      [0.6, 0.4]])

pi = stationary_distribution(P_weather)
print(f"Stationary distribution: Sunny={pi[0]:.4f}, Rainy={pi[1]:.4f}")
```

```
Stationary distribution: Sunny=0.6667, Rainy=0.3333
```

22.4.2 Verifying Convergence

We can verify the Ergodic Theorem numerically by powering the transition matrix:

```
# Confirm convergence: P^t rows all approach pi as t grows large
P_power = np.linalg.matrix_power(P_weather, 50)
print("P^50 (rows should all equal stationary distribution):")
print(np.round(P_power, 6))
```

```
P^50 (rows should all equal stationary distribution):
[[0.666667 0.333333]
 [0.666667 0.333333]]
```

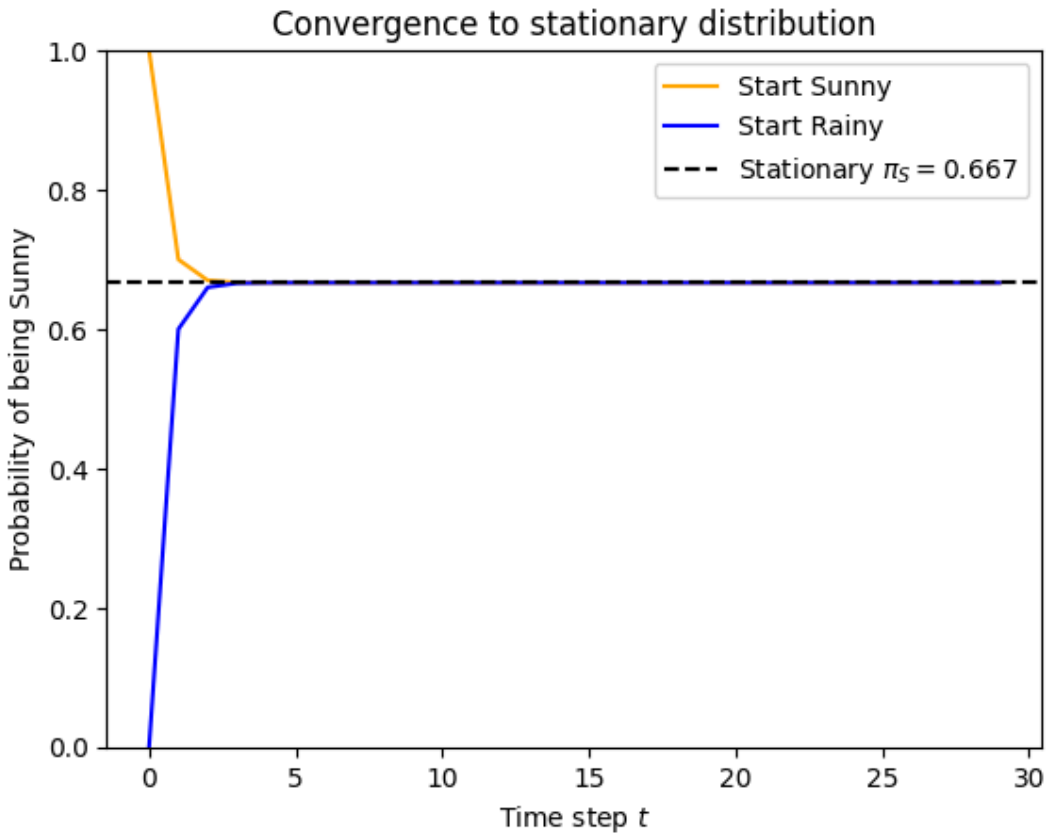
22.4.3 Stationary Distribution of a 3-State Chain

```
P_3 = np.array([[0.0, 0.6, 0.4],
                [0.3, 0.0, 0.7],
                [0.5, 0.5, 0.0]])

pi_3 = stationary_distribution(P_3)
print(f"Stationary distribution: {np.round(pi_3, 4)}")

# Verify: pi @ P should equal pi
print(f"Verification pi @ P: {np.round(pi_3 @ P_3, 4)}")
```

```
Stationary distribution: [0.2863 0.3524 0.3612]
Verification pi @ P: [0.2863 0.3524 0.3612]
```

22.4.4 Visualising Convergence from Different Starting States

```
import matplotlib.pyplot as plt

T = 30
pi_0_sunny = np.array([1.0, 0.0])  # start Sunny
pi_0_rainy = np.array([0.0, 1.0])  # start Rainy

dist_sunny = [pi_0_sunny @ np.linalg.matrix_power(P_weather, t) for t in
range(T)]
dist_rainy = [pi_0_rainy @ np.linalg.matrix_power(P_weather, t) for t in
range(T)]

plt.figure()
plt.plot([d[0] for d in dist_sunny], label="Start Sunny", color="orange")
plt.plot([d[0] for d in dist_rainy], label="Start Rainy", color="blue")
plt.axhline(pi[0], color="black", linestyle="--", label=f"Stationary $\
\pi_S={pi[0]:.3f}$")
plt.xlabel("Time step $t$")
plt.ylabel("Probability of being Sunny")
plt.title("Convergence to stationary distribution")
plt.legend()
plt.ylim(0, 1);
```



22.5 Notable Research

The theory of ergodic Markov chains builds on the foundational work of Andrei Markov himself, who first studied the long-run behaviour of dependent random sequences in the early twentieth century.

The modern treatment of ergodic chains and stationary distributions is presented in the classical reference (Kemeny & Snell, 1976), which establishes the Ergodic Theorem in the finite-state setting and its connections to the fundamental matrix theory of absorbing chains.

The application of ergodic theory to repeated games and direct reciprocity is central to the analysis of reactive strategies in the Prisoner's Dilemma, as explored in Direct Reciprocity. The computation of long-run payoffs from stationary distributions provides the mathematical foundation for Press and Dyson's influential analysis of memory-one strategies (Press & Dyson, 2012).

22.6 Conclusion

This appendix introduced **ergodic Markov chains** and the **Ergodic Theorem**, which guarantees convergence to a unique stationary distribution for chains that are irreducible and aperiodic. These tools are essential for analysing systems that cycle indefinitely rather than being absorbed.

The key concepts are summarised in Table 22.1.

Concept	Description
Irreducible Chain	Every state is reachable from every other state
Aperiodic Chain	No state is locked into returning only at multiples of a fixed period
Ergodic Chain	Irreducible and aperiodic; converges to a unique stationary distribution
Stationary Distribution	$\pi P = \pi$, $\sum \pi_i = 1$; gives long-run fraction of time in each state

Ergodic Theorem	$\pi^{(0)} P^t \to \pi$ for any initial distribution $\pi^{(0)}$

Table A: Summary of ergodic Markov chains

Important

The stationary distribution of an ergodic Markov chain provides the long-run time average of any observable quantity. This makes it the key tool for computing long-run payoffs in repeated games and evolutionary models with ongoing interaction, as opposed to absorbing chains where the relevant quantity is the probability of reaching each terminal state.

22.7 Solutions

Solution A: Solution to Exercise A3.1

1. $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

- **Irreducible?** Yes: from state 1 we reach state 2 in one step, and from state 2 we reach state 1 in one step.
- **Aperiodic?** No. The period of each state is $d = 2$, since return to state 1 is only possible in an even number of steps (2, 4, 6, ...). The diagonal of P is zero, and $P^2 = I$ has positive diagonal, confirming period 2.
- **Ergodic?** No (not aperiodic).

Since the chain is not ergodic, the Ergodic Theorem does not apply in its strong form. However, because the chain is irreducible, there is still a unique stationary distribution. Solving $\pi P = \pi$ with $\pi_1 + \pi_2 = 1$ gives $\pi_2 = \pi_1$ and hence $\pi = (1/2, 1/2)$. The long-run time average is $(1/2, 1/2)$, but the chain does not converge from an arbitrary starting point; it oscillates.

2. $P = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}$

- **Irreducible?** Yes: both off-diagonal entries are positive.
- **Aperiodic?** Yes: both diagonal entries are positive ($P_{11} = 0.5 > 0$ and $P_{22} = 0.7 > 0$), so each state has a self-loop and $d_i = 1$.
- **Ergodic?** Yes.

Stationary distribution. Solving $\pi P = \pi$:

$$0.5\pi_1 + 0.3\pi_2 = \pi_1 \implies 0.3\pi_2 = 0.5\pi_1 \implies \pi_1 = \frac{3}{5}\pi_2 \tag{N}$$

With $\pi_1 + \pi_2 = 1$:

$$\frac{3}{5}\pi_2 + \pi_2 = 1 \implies \pi_2 = \frac{5}{8}, \quad \pi_1 = \frac{3}{8}. \tag{O}$$

So $\pi = (3/8,\ 5/8) = (0.375,\ 0.625)$.

3. $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}$

- **Irreducible?** No. State 1 has $P_{11} = 1$ (it is absorbing). Once the chain is in state 1 it never reaches states 2 or 3. States 2 and 3 communicate with each other but not with state 1.
- **Aperiodic?** State 1 is aperiodic (self-loop). States 2 and 3 have period 1 since P^2 has positive diagonal entries. But irreducibility fails, so the question of chain-level aperiodicity is moot.

- **Ergodic?** No (not irreducible).

The chain has multiple closed communicating classes and is therefore not ergodic; there is no unique stationary distribution for any initial state.

```python
import numpy as np

def stationary_distribution(P):
    n = P.shape[0]
    A = (P - np.eye(n)).T
    A[-1, :] = 1.0
    b = np.zeros(n)
    b[-1] = 1.0
    return np.linalg.solve(A, b)

P2 = np.array([[0.5, 0.5], [0.3, 0.7]])
pi2 = stationary_distribution(P2)
print("Chain 2 stationary distribution:", np.round(pi2, 6))
# Verify
print("Verification pi @ P:", np.round(pi2 @ P2, 6))
```

```
Chain 2 stationary distribution: [0.375 0.625]
Verification pi @ P: [0.375 0.625]
```

Solution B: Solution to Exercise A3.2

Let $P = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}$ with $0 < a, b < 1$.

1. **Ergodicity.** Since $a > 0$ and $b > 0$, both off-diagonal entries are positive: state 1 can reach state 2 in one step and vice versa, so the chain is **irreducible**. The diagonal entries satisfy $1 - a \in (0, 1)$ and $1 - b \in (0, 1)$, so both states have self-loops and $d_i = 1$ for all i: the chain is **aperiodic**. Hence it is **ergodic** for any $a, b \in (0, 1)$.
2. **Stationary distribution.** The condition $\pi P = \pi$ gives:

$$\pi_1(1-a) + \pi_2 b = \pi_1 \implies \pi_1 a = \pi_2 b \implies \pi_1 = \frac{b}{a}\pi_2. \tag{P}$$

Using $\pi_1 + \pi_2 = 1$:

$$\frac{b}{a}\pi_2 + \pi_2 = 1 \implies \pi_2 \frac{a+b}{a} = 1 \implies \pi_2 = \frac{a}{a+b}. \tag{Q}$$

Therefore:

$$\pi_1 = \frac{b}{a+b}, \qquad \pi_2 = \frac{a}{a+b}. \tag{R}$$

1. **Condition for uniform stationary distribution.** We need $\pi_1 = \pi_2 = 1/2$:

$$\frac{b}{a+b} = \frac{1}{2} \Leftrightarrow 2b = a + b \Leftrightarrow a = b. \tag{S}$$

The stationary distribution is uniform if and only if $a = b$, the two transition rates are equal.

```python
import sympy as sym

a, b = sym.symbols("a b", positive=True)
P_sym = sym.Matrix([[1 - a, a], [b, 1 - b]])

pi1, pi2 = sym.symbols("pi_1 pi_2")
eqs = [
    sym.Eq(pi1 * (1 - a) + pi2 * b, pi1),
    sym.Eq(pi1 + pi2, 1),
]
sol = sym.solve(eqs, [pi1, pi2])
print("Stationary distribution:")
print("pi_1 =", sol[pi1])
print("pi_2 =", sol[pi2])
```

```
Stationary distribution:
pi_1 = b/(a + b)
pi_2 = a/(a + b)
```

```python
# Condition for pi_1 = pi_2
condition = sym.solve(sym.Eq(sol[pi1], sym.Rational(1, 2)), a)
print("Condition for uniform distribution: a =", condition)
```

```
Condition for uniform distribution: a = [b]
```

Solution C: Solution to Exercise A3.3

The weather chain from Motivating Example: Long-Run Behaviour in a Cycling System has:

$$P = \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \tag{T}$$

with $\alpha = 0.3$, $\beta = 0.6$ and stationary distribution $\pi = (2/3, 1/3)$.

1. **Computing $\pi^{(1)}$ and $\pi^{(2)}$ from $\pi^{(0)} = (1, 0)$.**

$$\begin{aligned} \pi^{(1)} &= \pi^{(0)} P = (1, 0) \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \\ &= (0.7,\ 0.3). \end{aligned} \tag{U}$$

$$\begin{aligned} \pi^{(2)} &= \pi^{(1)} P = (0.7, 0.3) \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \\ &= (0.7 \times 0.7 + 0.3 \times 0.6,\ 0.7 \times 0.3 + 0.3 \times 0.4) \\ &= (0.49 + 0.18,\ 0.21 + 0.12) \\ &= (0.67,\ 0.33). \end{aligned} \tag{V}$$

1. **Distance from the stationary distribution.** The stationary distribution is $\pi = (2/3, 1/3) \approx (0.6\overline{6}, 0.3\overline{3})$.

$$\begin{aligned} \| \pi^{(2)} - \pi \|_1 &= | \, 0.67 - 0.6\overline{6} \, | + | \, 0.33 - 0.3\overline{3} \, | \\ &= 0.003\overline{3} + 0.003\overline{3} \approx 0.0067. \end{aligned} \tag{W}$$

After only two steps, the distribution is within less than 1% of the stationary distribution in ℓ^1 distance. Convergence is very fast.

1. **Eigenvalues of P.** The characteristic polynomial of P is:

$$\begin{aligned} \det(P - \lambda I) &= (0.7 - \lambda)(0.4 - \lambda) - (0.3)(0.6) \\ &= \lambda^2 - 1.1\lambda + (0.28 - 0.18) \\ &= \lambda^2 - 1.1\lambda + 0.10. \end{aligned} \tag{X}$$

The roots are:

$$\begin{aligned} \lambda &= \frac{1.1 \pm \sqrt{1.21 - 0.4}}{2} = \frac{1.1 \pm \sqrt{0.81}}{2} \\ &= \frac{1.1 \pm 0.9}{2}. \end{aligned} \tag{Y}$$

So $\lambda_1 = 1$ and $\lambda_2 = 0.1$.

Verification: $|\,1 - \alpha - \beta\,| = |\,1 - 0.3 - 0.6\,| = |\,0.1\,| = 0.1 = |\,\lambda_2\,|$. ✓

The second eigenvalue governs the exponential rate of convergence: $|\,\lambda_2\,|^t = 0.1^t$, which decays rapidly (after just 2 steps, $|\,\lambda_2\,|^2 = 0.01$), consistent with the fast convergence observed above.

```
import numpy as np

P = np.array([[0.7, 0.3], [0.6, 0.4]])
pi_0 = np.array([1.0, 0.0])

pi_1 = pi_0 @ P
pi_2 = pi_1 @ P
print("pi^(1) =", pi_1)
print("pi^(2) =", pi_2)

pi_stationary = np.array([2/3, 1/3])
print("Stationary distribution:", np.round(pi_stationary, 6))
print("L1 distance after 2 steps:", np.sum(np.abs(pi_2 - pi_stationary)))
```

```
pi^(1) = [0.7 0.3]
pi^(2) = [0.67 0.33]
Stationary distribution: [0.666667 0.333333]
L1 distance after 2 steps: 0.006666666666666654
```

```
eigenvalues = np.linalg.eigvals(P)
print("Eigenvalues of P:", np.sort(eigenvalues)[::-1])
alpha, beta = 0.3, 0.6
print("|1 - alpha - beta| =", abs(1 - alpha - beta))
```

```
Eigenvalues of P: [1.  0.1]
|1 - alpha - beta| = 0.09999999999999998
```

Solution D: Solution to Exercise A3.4

Two tit-for-tat (TFT) players with $(p, q) = (1, 0)$ each cooperate if the opponent cooperated last round and defect if the opponent defected last round. The four states are $\{CC, CD, DC, DD\}$, where the first letter is player 1's action and the second is player 2's action in the previous round.

1. **Transition matrix.** Given current state (row player's last action, column player's last action):
    - From CC: TFT1 plays C (since opponent played C; $p = 1$) and TFT2 plays C (since opponent played C; $p' = 1$). Next state: CC.
    - From CD: TFT1 plays D (since opponent played D; $q = 0$) and TFT2 plays C (since opponent played C; $p' = 1$). Next state: DC.
    - From DC: TFT1 plays C (since opponent played C; $p = 1$) and TFT2 plays D (since opponent played D; $q' = 0$). Next state: CD.
    - From DD: TFT1 plays D (since opponent played D; $q = 0$) and TFT2 plays D (since opponent played D; $q' = 0$). Next state: DD.

The transition matrix (ordering states CC, CD, DC, DD) is:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{Z}$$

1. **Ergodicity.** The chain is **not ergodic**:
    - State CC is absorbing ($P_{CC,CC} = 1$). Once both cooperate, they cooperate forever.
    - State DD is absorbing ($P_{DD,DD} = 1$). Once both defect, they defect forever.
    - States CD and DC form a periodic communicating class with period 2 (they cycle: $CD \to DC \to CD \to \cdots$).
    - The chain is **reducible**: state CC cannot reach DD or the $\{CD, DC\}$ cycle, and vice versa.

Therefore the chain is neither irreducible nor aperiodic, hence not ergodic.

1. **Long-run distribution starting from CC.** Since CC is an absorbing state and the game starts in CC, the chain stays in CC with probability 1 for all time. The long-run distribution is:

$$\pi = (1, 0, 0, 0) \quad (\text{all probability in state } CC\ ). \tag{AA}$$

In terms of the game: two TFT players who start by cooperating will cooperate forever. The long-run payoff per round equals the cooperation payoff R.

```python
import numpy as np

# TFT vs TFT transition matrix over states {CC, CD, DC, DD}
P_tft = np.array([
    [1, 0, 0, 0],  # CC -> CC
    [0, 0, 1, 0],  # CD -> DC
    [0, 1, 0, 0],  # DC -> CD
    [0, 0, 0, 1],  # DD -> DD
], dtype=float)
```

```
print("Transition matrix P:")
print(P_tft)

# Starting from CC (index 0), pi^(0) = (1, 0, 0, 0)
pi_0 = np.array([1.0, 0.0, 0.0, 0.0])
print("\nAfter 10 steps (from CC):", pi_0 @ np.linalg.matrix_power(P_tft, 10))
```

```
Transition matrix P:
[[1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]]

After 10 steps (from CC): [1. 0. 0. 0.]
```

23. Appendix: Karush-Kuhn-Tucker Conditions

23.1 Motivating Example: Optimising doughnut distribution at a research retreat

At the annual **Mathematical Modelling Retreat**, a catering mix-up leaves the organisers with a limited supply of **12 gourmet doughnuts** and **8 litres of coffee**.

To ensure everyone gets a share, the organisers plan to slice the doughnuts into small portions and pour the coffee into individual cups of varying sizes.

The participants are divided into two groups:

- **Theoretical mathematicians**, who greatly enjoy strong coffee and appreciate a modest amount of sweet pastry.
- **Applied mathematicians**, who relish the pastries but are content with just enough coffee to stay alert.

Let:

- x_1 represent the number of *doughnut portions* allocated to the theorists
- x_2 represent the amount of *coffee (in litres)* allocated to the theorists

The applied group receives the remaining quantities:

- $(12 - x_1)$ doughnut portions
- $(8 - x_2)$ litres of coffee

The **total enjoyment** of the retreat is modelled as the sum of group satisfactions:

$$f(x_1, x_2) = \log(1 + x_1) + 2\sqrt{x_2}\log(1 + 12 - x_1) + 2\sqrt{8 - x_2} \tag{A}$$

subject to the feasibility constraints:

$$0 \leq x_1 \leq 12, \quad 0 \leq x_2 \leq 8 \tag{B}$$

The problem is now a continuous **convex optimisation problem**: although doughnuts and coffee are discrete in reality, the organisers can approximate an optimal allocation by dividing portions finely, making fractional allocations meaningful.

This is an ideal setting for the **Karush-Kuhn-Tucker (KKT) conditions**.

In the next section, we will formally introduce these conditions and use them to solve this doughnut-and-coffee dilemma.

23.2 Theory

23.2.1 Definition: The Karush-Kuhn-Tucker Conditions

Let $x \in \mathbb{R}^n$ be a vector of decision variables. We consider a general nonlinear programme of the form:

$$\begin{aligned} \text{Minimise} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{C}$$

- The function $f : \mathbb{R}^n \to \mathbb{R}$ is the objective.
- The functions g_i are inequality constraints.
- The functions h_j are equality constraints.

We seek a point x^* that satisfies all constraints and minimises $f(x)$.

Suppose f, g_i, and h_j are continuously differentiable, and suppose a constraint qualification holds at a local minimum x^*.

Then there exist Lagrange multipliers $\lambda_i \geq 0$ and $\mu_j \in \mathbb{R}$ such that the Karush-Kuhn-Tucker (KKT) conditions hold:

- **Stationarity**:

$$\nabla_x \mathcal{L}(x, \mu, \lambda) = \nabla_x f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{p} \mu_j \nabla h_j(x^*) = 0 \tag{D}$$

- **Primal feasibility**:

$$g_i(x^*) \leq 0, \quad h_j(x^*) = 0 \tag{E}$$

- **Dual feasibility**:

$$\lambda_i \geq 0 \tag{F}$$

- **Complementary slackness**:

$$\lambda_i g_i(x^*) = 0 \quad \text{for all } i \tag{G}$$

These conditions are **necessary** for local optimality, and under convexity assumptions, they are also **sufficient**.

Each of the conditions has an important role:

- **Stationarity**

At a local optimum, the gradient of the Lagrangian vanishes. This means that the direction of steepest descent of the objective is balanced by the gradients of the active constraints.

- **Primal feasibility**

The candidate solution must satisfy all constraints of the original problem: the inequality constraints must be nonpositive, and the equality constraints must be exactly satisfied.

- **Dual feasibility**

The Lagrange multipliers corresponding to inequality constraints must be nonnegative. These multipliers can be interpreted as shadow prices or marginal rates of improvement in the objective function per unit tightening of the respective constraints. A negative multiplier would suggest that tightening a constraint could worsen the objective, which contradicts optimality.

- **Complementary slackness**

For each inequality constraint, either the constraint is active (holds with equality) and its multiplier may be positive, or the constraint is inactive (strictly satisfied) and its multiplier must be zero. This ensures that inactive constraints do not influence the stationarity condition.

23.2.2 Example: Verifying the KKT conditions for the doughnut problem

Let us verify that $x_1 \approx 1.243$ and $x_2 \approx 6.869$ satisfy the KKT conditions for the doughnut problem.

The objective function is given by $f(x_1, x_2)$. The inequality constraints can be written as:

$$
\begin{aligned}
g_1(x) &= -x_1 \leq 0 \\
g_2(x) &= -x_2 \leq 0 \\
g_3(x) &= x_1 - 12 \leq 0 \\
g_4(x) &= x_2 - 8 \leq 0
\end{aligned}
\tag{H}
$$

There are no equality constraints in this problem, so there are no associated Lagrange multipliers μ.

The Lagrangian is:

$$
\mathcal{L}(x, \lambda) = -\lambda_1 x_1 + \lambda_2(x_1 - 12) - \lambda_3 x_2 + \lambda_4(x_2 - 8) + 2\sqrt{x_2}\log(13 - x_1) + 2\sqrt{8 - x_2} + \log(x_1 + 1)
\tag{I}
$$

We now check whether $x \approx (1.243, 6.869)$ satisfies the KKT conditions.

Stationarity:

We compute the partial derivatives of the Lagrangian:

$$
\begin{aligned}
\frac{\partial \mathcal{L}(x, \lambda)}{\partial x_1} &= -\lambda_1 + \lambda_2 - \frac{2\sqrt{x_2}}{13 - x_1} + \frac{1}{x_1 + 1} \\
\frac{\partial \mathcal{L}(x, \lambda)}{\partial x_2} &= -\lambda_3 + \lambda_4 - \frac{1}{\sqrt{8 - x_2}} + \frac{\log(13 - x_1)}{\sqrt{x_2}}
\end{aligned}
\tag{J}
$$

Primal feasibility, dual feasibility, and complementary slackness:

Substituting $x = (1.243, 6.869)$ into the constraints gives:

$$
\begin{aligned}
g_1(x) &= -1.243 < 0 \\
g_2(x) &= -6.869 < 0 \\
g_3(x) &= -10.757 < 0 \\
g_4(x) &= -1.131 < 0
\end{aligned}
\tag{K}
$$

Since all inequality constraints are strictly satisfied, complementary slackness implies that $\lambda_i = 0$ for all i. This also satisfies dual feasibility.

Verifying stationarity:

Substituting $x = (1.243, 6.869)$ and $\lambda_i = 0$ into the stationarity conditions gives:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x_1} \Big|_{x, \lambda = 0} &= -\frac{2\sqrt{6.869}}{13 - 1.243} + \frac{1}{1.243 + 1} \approx 0 \\
\frac{\partial \mathcal{L}}{\partial x_2} \Big|_{x, \lambda = 0} &= -\frac{1}{\sqrt{8 - 6.869}} + \frac{\log(13 - 1.243)}{\sqrt{6.869}} \approx 0
\end{aligned}
\tag{L}
$$

Both partial derivatives are approximately zero, confirming that stationarity holds.

As $f(x_1, x_2)$ is a sum of concave functions being maximised (or equivalently, a convex minimisation problem), the KKT conditions are both necessary and sufficient. We can conclude that $x \approx (1.243, 6.869)$ is indeed the global optimum.

23.3 Exercises

23.3.1 Exercise 1: Understanding KKT components

Consider the following optimisation problem:

$$\begin{aligned} \text{Minimise} \quad & f(x, y) = x^2 + y^2 \\ \text{subject to} \quad & x + y \geq 2 \\ & x \geq 0, \quad y \geq 0 \end{aligned} \tag{M}$$

1. Rewrite the problem in standard form for KKT analysis (i.e., with all constraints as $g_i(x) \leq 0$).
2. Write down the KKT conditions symbolically.
3. Find all KKT points and identify whether they correspond to a global minimum.

23.3.2 Exercise 2: Symbolic derivation of KKT conditions

Consider the problem:

$$\begin{aligned} \text{Maximise} \quad & \log(x) + \log(1 - x) \\ \text{subject to} \quad & 0.1 \leq x \leq 0.9 \end{aligned} \tag{N}$$

1. Show that the objective function is concave on the interval $(0, 1)$.
2. Derive the KKT conditions for this problem.
3. Solve for the optimal value of x and the associated Lagrange multipliers.

23.4 Programming

23.4.1 Solving constrained optimisation problems with Scipy

Scipy has functionality to solve constrained optimisation problems.

```
import numpy as np
import scipy.optimize

def objective(x):
    x1, x2 = x
    return -(np.log(1 + x1) + 2 * np.sqrt(x2) * np.log(1 + 12 - x1) + 2 *
np.sqrt(8 - x2))

def g_1(x):
    return x[0]

def g_2(x):
    return x[1]

def g_3(x):
    return 12 - x[0]

def g_4(x):
    return 8 - x[1]

constraints = [
    {'type': 'ineq', 'fun': g_1},
    {'type': 'ineq', 'fun': g_2},
    {'type': 'ineq', 'fun': g_3},
    {'type': 'ineq', 'fun': g_4},
]

res = scipy.optimize.minimize(objective, [6, 4], constraints=constraints)
res
```

```
     message: Optimization terminated successfully
     success: True
      status: 0
         fun: -15.852821854736256
           x: [ 1.243e+00  6.869e+00]
         nit: 10
         jac: [ 3.338e-06  3.695e-06]
        nfev: 31
        njev: 10
 multipliers: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
```

23.5 Notable Research

The Karush-Kuhn-Tucker conditions were first described by Karush in (Karush, 1939) but this was unpublished and widely unknown until Kuhn and Tucker presented the conditions at a mathematics symposium (Kuhn & Tucker, 1951). Some text book still refer to the conditions as the Kuhn-Tucker conditions but most, rightly, refer to them in a way that rightfully respects all the authors.

Before (Kuhn & Tucker, 1951) generalisation of the conditions that holds in more complex cases even when the feasible region is not well behaved was published in (John, 1948).

The Karush-Kuhn-Tucker conditions provide a way to check stability or indeed optimality but the collection of methods that finds potential optima is referred to as interior point method (Boyd & Vandenberghe, 2004).

23.6 Conclusion

The Karush-Kuhn-Tucker (KKT) conditions provide a unified framework to characterise optimal solutions to constrained optimisation problems. They extend the method of Lagrange multipliers to include inequality constraints, introducing complementary slackness and dual feasibility as key components.

Table 23.1 summarises the key concepts introduced in this appendix.

Condition	Meaning
Stationarity	The objective's gradient is balanced by the gradients of active constraints
Primal feasibility	The solution satisfies all original constraints
Dual feasibility	Lagrange multipliers for inequality constraints are nonnegative
Complementary slackness	Inactive constraints have zero multipliers; active constraints may have positive multipliers

Table A: Summary of the Karush-Kuhn-Tucker conditions

The KKT conditions are **necessary** for optimality when constraint qualifications hold, and **sufficient** for optimality when the problem is convex. In this way, they serve both as a *diagnostic tool* to verify candidate solutions, and as a *theoretical foundation* for many modern optimisation algorithms.

Important

In unconstrained optimisation, optimality is characterised by setting the derivative to zero. With equality constraints, we introduce the Lagrangian and set its gradient to zero. With inequality constraints, the Karush-Kuhn-Tucker conditions extend this further, adding complementary slackness and nonnegative multipliers to capture which constraints are active at the optimum.

23.7 Solutions

23.7.1 Solution to Exercise 1: Understanding KKT Components

The problem is:

$$\begin{aligned} \text{Minimise} \quad & f(x,y) = x^2 + y^2 \\ \text{subject to} \quad & x + y \geq 2 \\ & x \geq 0, \quad y \geq 0 \end{aligned} \tag{O}$$

1. **Standard form.** We rewrite all constraints as $g_i \leq 0$:

$$\begin{aligned} g_1(x,y) &= -(x+y-2) = 2 - x - y \leq 0 \\ g_2(x,y) &= -x \leq 0 \\ g_3(x,y) &= -y \leq 0 \end{aligned} \tag{P}$$

There are no equality constraints (h_j).

1. **KKT conditions.** With Lagrange multipliers $\lambda_1, \lambda_2, \lambda_3 \geq 0$, the Lagrangian is:

$$\mathcal{L}(x,y,\lambda) = x^2 + y^2 + \lambda_1(2 - x - y) - \lambda_2 x - \lambda_3 y. \tag{Q}$$

Stationarity:

$$\frac{\partial \mathcal{L}}{\partial x} = 2x - \lambda_1 - \lambda_2 = 0 \qquad \Rightarrow \qquad 2x = \lambda_1 + \lambda_2 \tag{R}$$

$$\frac{\partial \mathcal{L}}{\partial y} = 2y - \lambda_1 - \lambda_3 = 0 \qquad \Rightarrow \qquad 2y = \lambda_1 + \lambda_3 \tag{S}$$

Primal feasibility:

$$2 - x - y \leq 0, \quad -x \leq 0, \quad -y \leq 0. \tag{T}$$

Dual feasibility:

$$\lambda_1, \lambda_2, \lambda_3 \geq 0. \tag{U}$$

Complementary slackness:

$$\lambda_1(2 - x - y) = 0, \quad \lambda_2(-x) = 0, \quad \lambda_3(-y) = 0. \tag{V}$$

1. **Finding the KKT point.** The objective $f(x,y) = x^2 + y^2$ is strictly convex, so any KKT point is the unique global minimum.

The unconstrained minimum is $(0,0)$, which violates $x + y \geq 2$. We therefore expect the constraint $g_1 = 0$ to be active, i.e., $x + y = 2$. By symmetry of the objective, the minimum on the line $x + y = 2$ with $x, y \geq 0$ occurs at $x = y = 1$.

Verify the KKT conditions at $(1,1)$:

Since $x = 1 > 0$ and $y = 1 > 0$, complementary slackness gives $\lambda_2 = \lambda_3 = 0$. Since $g_1 = 0$ (active), λ_1 may be nonzero.

From stationarity:

$$2(1) = \lambda_1 + 0 \Longrightarrow \lambda_1 = 2. \tag{W}$$

Check: $2y = 2(1) = 2 = \lambda_1 + \lambda_3 = 2 + 0$ ✓.

All KKT conditions are satisfied with $\lambda_1 = 2$, $\lambda_2 = \lambda_3 = 0$.

The global minimum is at $(x^*, y^*) = (1, 1)$ with $f(1, 1) = 2$.

```
import numpy as np
import scipy.optimize

def objective(v):
    return v[0]**2 + v[1]**2

constraints = [
    {'type': 'ineq', 'fun': lambda v: v[0] + v[1] - 2},
    {'type': 'ineq', 'fun': lambda v: v[0]},
    {'type': 'ineq', 'fun': lambda v: v[1]},
]

res = scipy.optimize.minimize(objective, [1, 1], constraints=constraints)
print("Optimal solution:", res.x)
print("Optimal value:", res.fun)
```

```
Optimal solution: [1. 1.]
Optimal value: 2.0
```

23.7.2 Solution to Exercise 2: Symbolic Derivation of KKT Conditions

The problem is:

$$\begin{aligned} &\text{Maximise} && \log(x) + \log(1 - x) \\ &\text{subject to} && 0.1 \leq x \leq 0.9 \end{aligned} \tag{X}$$

1. **Concavity of the objective.** Let $h(x) = \log(x) + \log(1 - x)$ on $(0, 1)$. Computing the second derivative:

$$h'(x) = \frac{1}{x} - \frac{1}{1 - x}, \qquad h''(x) = -\frac{1}{x^2} - \frac{1}{(1 - x)^2} < 0 \quad \forall x \in (0, 1). \tag{Y}$$

Since $h''(x) < 0$ on $(0, 1)$, the function is strictly concave on this interval.

1. **KKT conditions.** We convert to minimisation of $-h(x)$ and write the constraints in standard form $g_i(x) \leq 0$:

$$g_1(x) = 0.1 - x \leq 0, \qquad g_2(x) = x - 0.9 \leq 0. \tag{Z}$$

The Lagrangian for the minimisation of $-\log(x) - \log(1 - x)$ is:

$$\mathcal{L}(x, \lambda) = -\log(x) - \log(1 - x) + \lambda_1(0.1 - x) + \lambda_2(x - 0.9). \tag{AA}$$

Stationarity:

$$\frac{d\mathcal{L}}{dx} = -\frac{1}{x} + \frac{1}{1 - x} - \lambda_1 + \lambda_2 = 0. \tag{AB}$$

Primal feasibility: $0.1 - x \leq 0$ and $x - 0.9 \leq 0$.

Dual feasibility: $\lambda_1, \lambda_2 \geq 0$.

Complementary slackness: $\lambda_1(0.1 - x) = 0$ and $\lambda_2(x - 0.9) = 0$.

1. **Solving for the optimal x.**

The unconstrained maximum of $h(x)$ satisfies $h'(x) = 0$:

$$\frac{1}{x} = \frac{1}{1-x} \implies x = 1 - x \implies x = \frac{1}{2}. \tag{AC}$$

Since $x^* = 1/2$ lies strictly inside $[0.1, 0.9]$, both constraints are inactive: $g_1(1/2) = 0.1 - 0.5 = -0.4 < 0$ and $g_2(1/2) = 0.5 - 0.9 = -0.4 < 0$.

Complementary slackness therefore requires $\lambda_1 = \lambda_2 = 0$.

Verifying stationarity with $x = 1/2$, $\lambda_1 = \lambda_2 = 0$:

$$-\frac{1}{1/2} + \frac{1}{1-1/2} - 0 + 0 = -2 + 2 = 0. \quad \checkmark \tag{AD}$$

All KKT conditions are satisfied. Since the problem is concave (equivalently, we are minimising a convex function), $x^* = 1/2$ is the global maximum with $\lambda_1 = \lambda_2 = 0$.

The maximum value is $h(1/2) = \log(1/2) + \log(1/2) = -2\log 2 \approx -1.386$.

```python
import numpy as np
import scipy.optimize

def neg_objective(x):
    return -(np.log(x[0]) + np.log(1 - x[0]))

constraints = [
    {'type': 'ineq', 'fun': lambda x: x[0] - 0.1},
    {'type': 'ineq', 'fun': lambda x: 0.9 - x[0]},
]

res = scipy.optimize.minimize(neg_objective, [0.5], constraints=constraints)
print("Optimal x:", res.x[0])
print("Maximum value:", -res.fun)
print("Lambda_1 (should be 0):", 0.0, "(constraint inactive)")
print("Lambda_2 (should be 0):", 0.0, "(constraint inactive)")
```

```
Optimal x: 0.5
Maximum value: -1.3862943611198906
Lambda_1 (should be 0): 0.0 (constraint inactive)
Lambda_2 (should be 0): 0.0 (constraint inactive)
```

```python
import sympy as sym

x = sym.Symbol("x")
h = sym.log(x) + sym.log(1 - x)
h_prime = sym.diff(h, x)
h_double_prime = sym.diff(h_prime, x)

print("h'(x) =", h_prime)
print("h''(x) =", sym.simplify(h_double_prime))
print("Unconstrained optimum:", sym.solve(h_prime, x))
print("h(1/2) =", h.subs(x, sym.Rational(1, 2)))
```

```
h'(x) = -1/(1 - x) + 1/x
h''(x) = -1/(x - 1)**2 - 1/x**2
Unconstrained optimum: [1/2]
h(1/2) = -2*log(2)
```

24. Appendix: Integer Pivoting

24.1 Motivating Example: Two Views of a Polytope

Consider the following set of points:

$$\mathcal{P} = \left\{ \sum_{i=1}^{4} \lambda_i v_i \;\middle|\; \begin{array}{c} \sum_{i=1}^{4} \lambda_i = 1 \\ \lambda_i \geq 0 \quad \text{for all } i \\ v_i \in \{(0,0), (1/3,0), (1/4,1/4), (0,1/3)\} \end{array} \right\} \tag{A}$$

This set $\mathcal{P}$ is a **polytope**, defined as the **convex hull** of a finite set of vertices:

$$V = \{(0,0),\ (1/3,0),\ (1/4,1/4),\ (0,1/3)\}. \tag{B}$$

A visualization is provided in Figure 24.1.

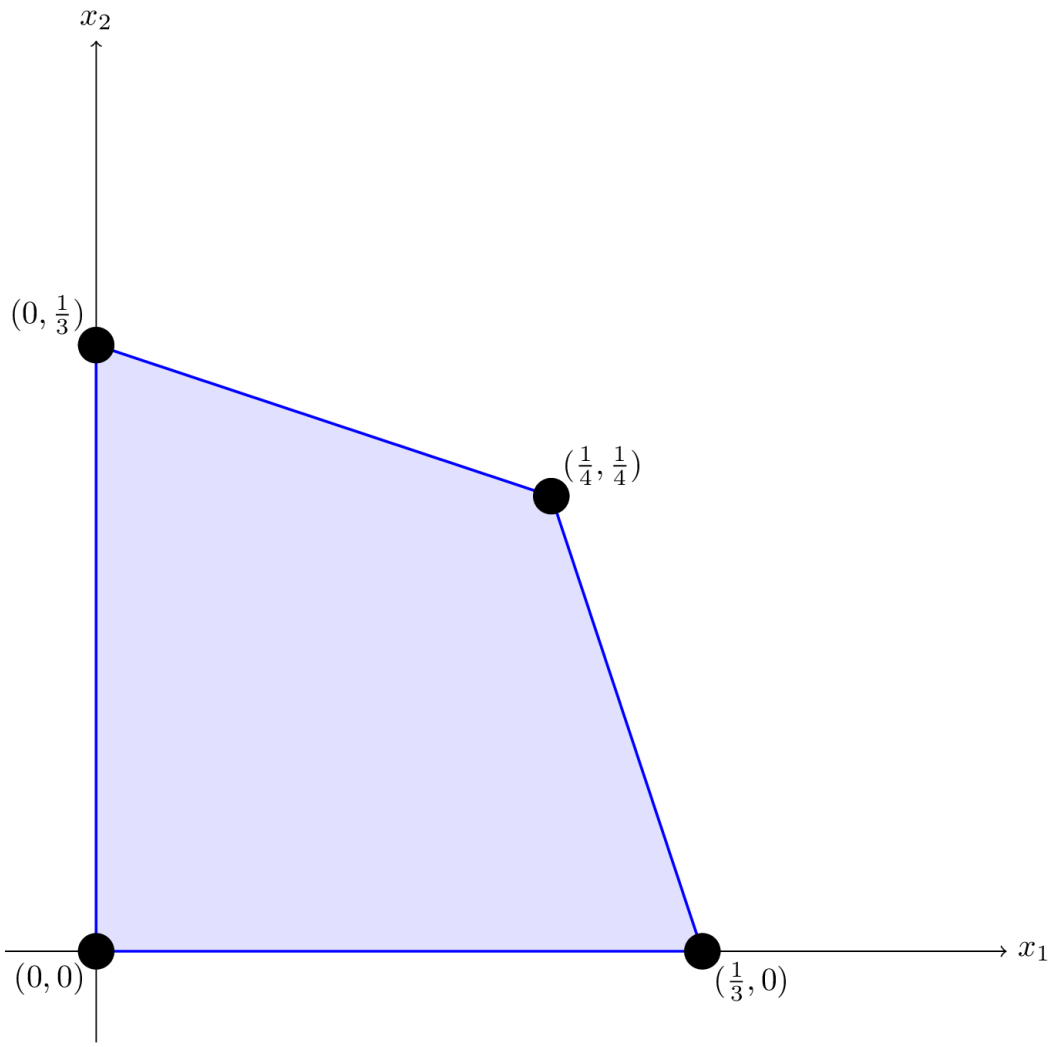


Figure A: The two-dimensional polytope defined by (24.1), shown as the convex hull of four points.

Polytopes can also be equivalently defined as the **intersection of half- spaces**, that is, bounded regions formed by linear inequalities. This alternate view is illustrated in Figure 24.2.

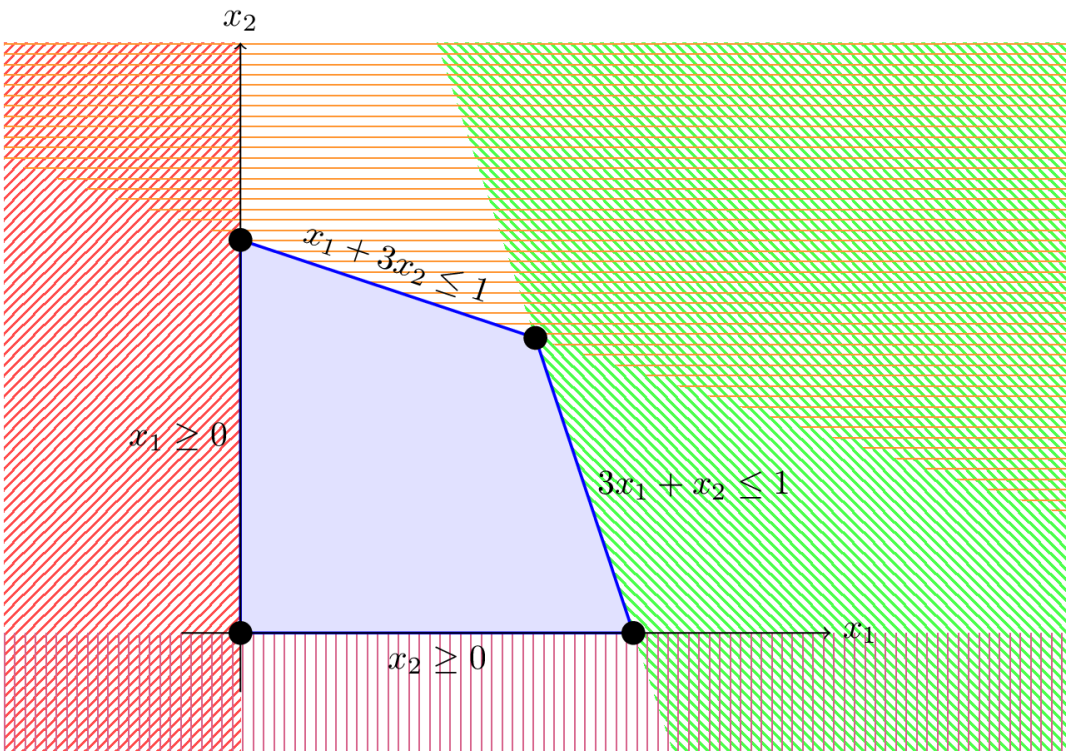


Figure B: The same polytope shown in Figure 24.1, represented as the intersection of linear inequalities.

This second definition corresponds to:

$$\mathcal{P} = \left\{ x \in \mathbb{R}^2 \mid \begin{array}{c} x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 + 3x_2 \leq 1 \\ 3x_1 + x_2 \leq 1 \end{array} \right\} \tag{C}$$

The inequalities in (24.3) define supporting half-spaces that together enclose the polytope.

Polytopes appear throughout mathematics, particularly in **optimisation**, where they form the feasible regions of linear programs. These connections are central to game theory, especially in the study of zero-sum games and general games.

While this example lives in $\mathbb{R}^2$, the techniques extend to **higher-dimensional polytopes**. The goal of this chapter is to develop tools for moving from **vertex to vertex** in such polytopes, an essential idea in both mathematical optimisation and algorithmic game theory.

24.2 Theory

24.2.1 Definition: Polytope as a Convex Hull

Let $V \subseteq \mathbb{R}^K$ be a finite set of points. A **polytope** $\mathcal{P}$ is the set of all **convex combinations** of points in V:

$$\mathcal{P} = \left\{ \sum_{i=1}^{|V|} \lambda_i v_i \in \mathbb{R}^K \mid \sum_{i=1}^{|V|} \lambda_i = 1,\ \lambda_i \geq 0,\ v_i \in V \right\} \tag{D}$$

24.2.2 Definition: Polytope as an Intersection of Half-Spaces

Let $M \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A **polytope** $\mathcal{P}$ can also be defined as the set of points satisfying a system of linear inequalities:

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Mx \leq b\} \tag{E}$$

Important

All polytopes can be represented either as a convex hull or as an intersection of half-spaces.

The representation of a polytope as the intersection of half-spaces via $Mx \leq b$ will allow us to work efficiently with large systems.

24.2.3 Definition: Slack Variable

Given a system of linear inequalities of the form $Mx \leq b$, a **slack variable** transforms each inequality into an equality by accounting for the difference between the left- and right-hand sides.

Formally, for each row i, the inequality

$$(Mx)_i \leq b_i \tag{F}$$

can be rewritten as

$$(Mx)_i + s_i = b_i \quad \text{with} \quad s_i \geq 0, \tag{G}$$

where $s_i \geq 0$ is the **slack variable** corresponding to the i-th inequality.

24.2.4 Definition: Tableau Representation of Vertices

For $M \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, given a system of linear inequalities:

$$Mx \leq b \tag{H}$$

This system of linear inequalities can be written as a system of equalities by adding slack variables to give a modified linear system of equalities:

$$\bar{M}\bar{x} = b \tag{I}$$

Where $\bar{M} \in \mathbb{R}^{m \times m+n}$ and $\bar{x} \in \mathbb{R}^{m+1}$ includes coefficients for the added slack variables:

$$\bar{M} = \begin{pmatrix} M & I_m \end{pmatrix} \qquad \bar{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ s_1 \\ \vdots \\ s_m \end{pmatrix} \tag{J}$$

Given the system $\bar{M}\bar{x} = b$ a tableau is an extended matrix of the form:

$$\begin{pmatrix} \bar{M} & I_m & b \end{pmatrix} \tag{K}$$

This augmented matrix of the underlying linear system is used to represent points in Polytopes. Through elementary row operations which will be called pivots we can move from one vertex to another.

24.2.5 Example: Tableaux for the Motivating Example

The linear system for the Motivating Example is:

$$\begin{aligned} x_1 + 3x_2 &\leq 1 \\ 3x_1 + x_2 &\leq 1 \end{aligned} \tag{L}$$

Note

We omit the non negativity inequality.

This corresponds to:

$$M = \begin{pmatrix} 1 & 3 \\ 3 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{M}$$

We can write a Tableau that represents the equivalent system of equations $\bar{M}\bar{x} = b$:

$$T^{(1)} = \begin{pmatrix} 1 & 3 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{N}$$

Tableaux are augmented matrices for linear systems, it is possible to do elementary row operations on them that do not modify the underlying system. We can also derive alternative tableaux from this example:

Multiplying $T^{(1)}$'s row 1 by 3 and subtracting row 2 gives:

$$T^{(2)} = \begin{pmatrix} 0 & 8 & 3 & -1 & 2 \\ 3 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{O}$$

Multiplying $T^{(2)}$'s row 2 by 8 and subtracting row 1 gives:

$$T^{(3)} = \begin{pmatrix} 0 & 8 & 3 & -1 & 2 \\ 24 & 0 & -3 & 9 & 6 \end{pmatrix} \tag{P}$$

Multiplying $T^{(3)}$'s row 1 by 9 and adding row 2 gives:

$$T^{(4)} = \begin{pmatrix} 24 & 72 & 24 & 0 & 24 \\ 24 & 0 & -3 & 9 & 6 \end{pmatrix} \tag{Q}$$

There are other tableaux that correspond to the same systems of equations but we next explore how tableaux correspond to vertices of polytopes. To do this we need a new definition:

24.2.6 Definition: Basic variables

A basic variable of a tableau corresponds to a column that is linearly independent from the others.

24.2.7 Example: Basic variables for the different Tableaux of the Motivating Example

The basic variables for:

- The tableau (24.14) are the two slack variables s_1 and s_2.
- The tableau (24.15) are x_1 and s_1.
- The tableau (24.16) are x_1 and x_2.
- The tableau (24.17) are x_2 and s_2.

Thus, the non-basic variables for:

- The tableau (24.14) are x_1 and x_2.
- The tableau (24.15) are x_2 and s_2.
- The tableau (24.16) are s_1 and s_2.
- The tableau (24.17) are x_1 and s_1.

24.2.8 Equivalence between a tableau and a vertex

A tableau represents the constraints and feasible region of a polytope. A given tableau represents a vertex of the polytope.

To obtain a vertex from the tableau:

- Set the non basic variables to be 0;
- Solve the remaining system of equations.

24.2.9 Example: equivalence between tableaux and vertices

1. For (24.14): we set the non-basic variables to 0:

$$x_1 = 0 \qquad x_2 = 0 \tag{R}$$

The remaining linear system from the tableau is:

$$\begin{aligned} s_1 &= 1 \\ s_2 &= 1 \end{aligned} \tag{S}$$

Note that we do not need to solve this remaining linear system as setting the non-basic variables to 0 immediately gives us the vertex: $(x_1, x_2) = (0, 0)$ (the origin of Figure 24.1).

1. For (24.15): we set the non-basic variables to 0:

$$x_2 = 0 \qquad s_2 = 0 \tag{T}$$

The remaining linear system from the tableau is:

$$\begin{aligned} 3x_1 &= 1 \\ 3s_1 &= 2 \end{aligned} \tag{U}$$

Solving this gives: $(x_1, x_2) = (1/3, 0)$ (the bottom right vertex of Figure 24.1).

1. For (24.16): we set the non-basic variables to 0:

$$s_1 = 0 \qquad s_2 = 0 \tag{V}$$

The remaining linear system from the tableau is:

$$\begin{aligned} 24x_1 &= 6 \\ 8x_2 &= 2 \end{aligned} \tag{W}$$

Solving this gives: $(x_1, x_2) = (1/4, 1/4)$ (the top right vertex of Figure 24.1).

1. For (24.17): we set the non-basic variables to 0:

$$x_1 = 0 \qquad s_1 = 0 \tag{X}$$

The remaining linear system from the tableau is:

$$\begin{aligned} 72x_2 &= 24 \\ 9s_2 &= 6 \end{aligned} \tag{Y}$$

Solving this gives: $(x_1, x_2) = (0, 1/3)$ (the top left vertex of Figure 24.1).

We have recovered all four vertices from the 4 tableaux of Example: Tableaux for the Motivating Example. Note that in that example through the process of applying elementary row operations to the tableaux we moved across the edges of the polytope as shown in Figure 24.3.

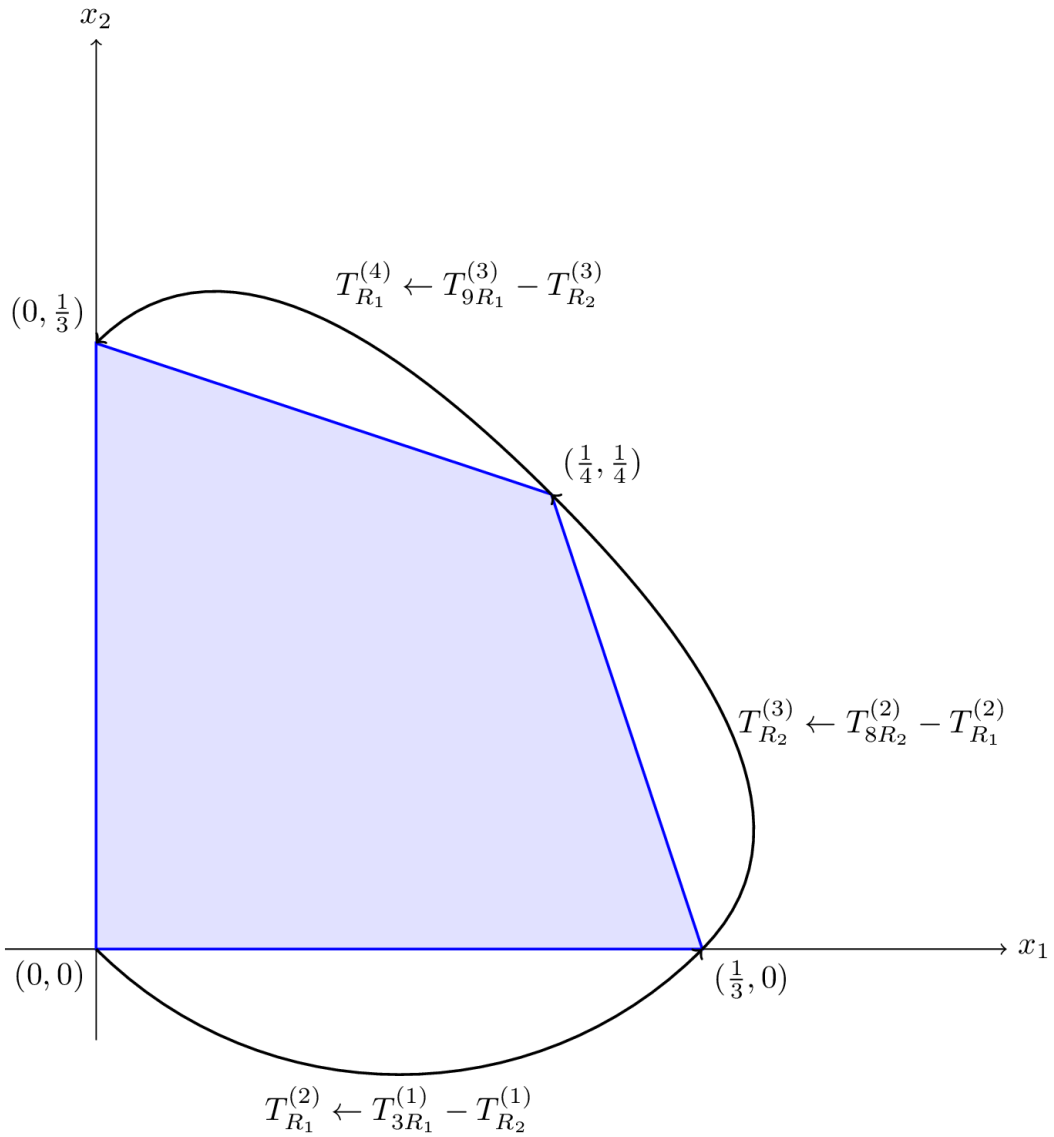


Figure C: The two-dimensional polytope defined by (24.1), with arrows showing the process of moving along the edges. The various row operations are written as short hand.

This process corresponds to making a non-basic variable basic and carefully choosing which basic variable to make non-basic.

24.2.10 Definition: Integer Pivoting

Pivoting is the process of updating a tableau by selecting one basic variable to leave the basis and one non-basic variable to enter it. This corresponds to performing row operations to rewrite the system so that the new basic variable appears with coefficient 1 in its row and 0 in all others.

In **integer pivoting**, we perform this process using only integer-preserving row operations, such as adding or subtracting integer multiples of rows. This is useful for maintaining exact arithmetic and geometric intuition in discrete settings.

Suppose you are given a tableau representing a system of equations and you choose a non-basic variable x_j to enter the basis.

Note

How a particular variable x_j is chosen is important and depends on the context. Two different approaches will be considered in chapters Zero-Sum Games and Best response polytopes.

The goal is to perform a **pivot** so that x_j becomes basic and one of the current basic variables is removed from the basis.

The pivot is carried out using the following steps:

1. **Identify the pivot column** Select the column of the tableau corresponding to x_j.
2. **Select the pivot row minimum ratio test** For each row i where the coefficient $T_{ij} > 0$, compute the ratio:

$$\frac{b_i}{T_{ij}} \tag{Z}$$

(where b_i corresponds to the final column of T.)

Choose the row r with the **smallest non-negative ratio**. The basic variable in this row will leave the basis.

1. **Eliminate the pivot column in all other rows** For each row $i \neq r$, if necessary multiply it by a suitable **integer multiplier** and then subtract suitable **integer multiples** of row r so that the entry in column j becomes zero.

After these steps, the tableau reflects a new basic solution where x_j is basic, and one previous basic variable has become non-basic.

Attention

The mimimum ratio test corresponds to finding which of the inequalities will become "tight" first.

For example for (24.15) recall:

$$T^{(1)} = \begin{pmatrix} 1 & 3 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{AA}$$

which has non basic variables x_1 and x_2. We have a choice of letting either of those two variables becoming basic. Let us choose x_2 to become basic. This requires us to choose which row will gain a 0 in the second column.

This choice corresponds to choosing which of the following two constraints becomes a tight equality first:

$$\begin{aligned} x_1 + 3x_2 &\leq 1 \\ 3x_1 + x_2 &\leq 1 \end{aligned} \tag{AB}$$

or equivalently:

$$\begin{aligned} x_1 + 3x_2 + s_1 &= 1 \\ 3x_1 + x_2 + s_2 &= 1 \end{aligned} \tag{AC}$$

as we want x_2 to enter while x_1 remains 0 this yields:

$$\begin{aligned} 3x_2 + s_1 &= 1 \\ x_2 + s_2 &= 1 \end{aligned} \tag{AD}$$

Either s_1 or s_2 must become 0 as shown in Figure 24.4. So either:

$$x_2 = 1/3$$
$$x_2 = 1 \tag{AE}$$

Note that if $x_2 = 1$ then the first constraint is not valid which is what the minimum ratio test is checking. The minimum ratio test would ensure we pivot on the first row.

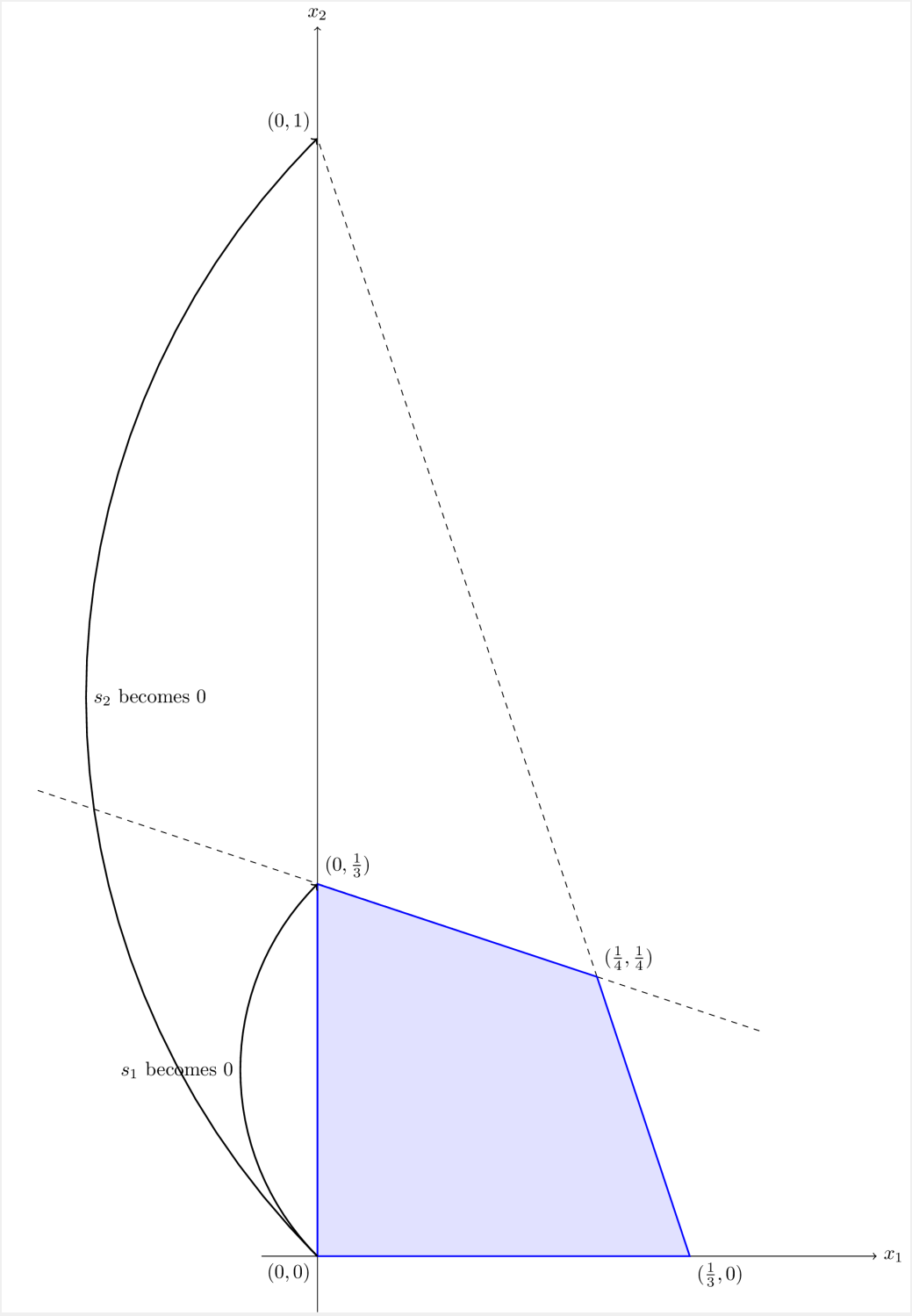


Figure D: Choosing which variable to make 0 which corresponds to it become basic. Given a choice of making x_2 non-basic, we move along the corresponding edge until we arrive at the first inequality constraint. This is equivalent to the minimum ratio test.

24.2.11 Example: Moving from $T^{(1)}$ to $T^{(2)}$

Let us start at:

$$T^{(1)} = \begin{pmatrix} 1 & 3 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{AF}$$

The non-basic variables are x_1 and x_2. Let us choose to let x_2 become basic. This corresponds to the second column of the tableau.

Next we use the minimum ratio test:

1. The ratio for the first row: $\frac{1}{3}$
2. The ratio for the second row: $\frac{1}{1}$

Both of these are non-negative and the minimum value is obtained in the first row. Thus we pivot on the first row making the value in the second column and "all other rows" (in this case just the second row) 0.

Multiplying $T^{(1)}$'s row 2 by 3 and subtracting row 1 gives:

$$T^{(2)} = \begin{pmatrix} 1 & 3 & 1 & 0 & 1 \\ 8 & 0 & -1 & 3 & 2 \end{pmatrix} \tag{AG}$$

Using Equivalence between a tableau and a vertex we set the basic variables to 0:

$$x_1 = 0 \qquad s_1 = 0 \tag{AH}$$

The remaining linear system from the tableau is:

$$\begin{aligned} 3x_2 &= 1 \\ 3s_2 &= 2 \end{aligned} \tag{AI}$$

Solving this gives: $(x_1, x_2) = (0, 1/3)$.

24.3 Exercises

24.3.1 Exercise: Polytope Representation

For each of the following sets of vertices:

- Draw the corresponding polytope.
- Write the polytope as the intersection of a set of halfspaces (i.e. inequalities).

1. $V = \{(0,0), (0,1), (1,0), (1,1)\}$
2. $V = \left\{(0,0), \left(0, \frac{1}{4}\right), \left(1, \frac{2}{3}\right), \left(2, \frac{1}{5}\right)\right\}$
3. $V = \left\{(0,0), \left(0, \frac{1}{4}\right), \left(1, \frac{2}{3}\right), \left(2, \frac{1}{5}\right), (1,0)\right\}$

24.3.2 Exercise: Basic and Non-Basic Variables

For each of the following tableaux, identify the **basic** and **non-basic** variables:

1.

$$T^{(a)} = \begin{pmatrix} 1 & 0 & 5 & 1 & 1 \\ 0 & 1 & 4 & 9 & 1 \end{pmatrix} \tag{AJ}$$

1.

$$T^{(b)} = \begin{pmatrix} 4 & 8 & 1 & 0 & 1 \\ 8 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{AK}$$

24.3.3 Exercise: Integer Pivoting

For the tableaux in Exercise: Basic and Non-Basic Variables:

For each **non-basic variable**, perform one step of integer pivoting:

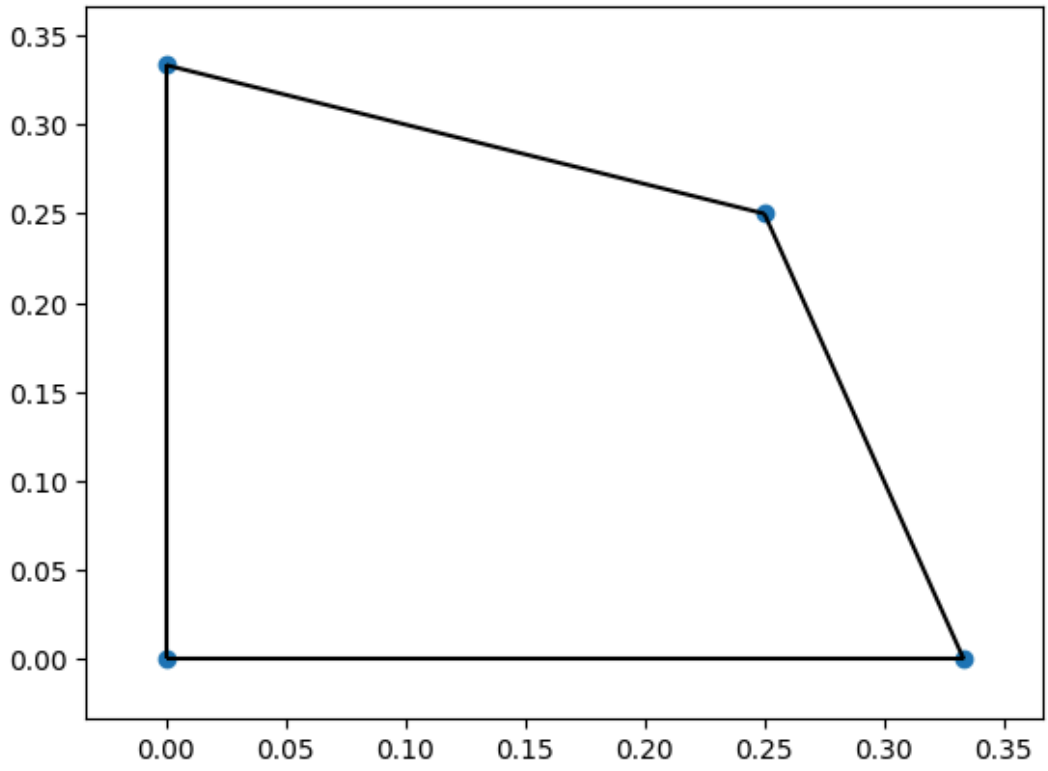
1. Identify the pivot column;
2. Perform the **minimum ratio test** to determine the pivot row;
3. Execute the required **row operations** to complete the pivot.

24.4 Programming

24.4.1 Using SciPy to obtain the convex hull

The SciPy library has functionality to obtain the convex hull of a set of vertices.

```
import scipy.spatial
import numpy as np

V = [np.array([0, 0]), np.array([1 / 3, 0]), np.array([1 / 4, 1 / 4]),
np.array([0, 1 / 3])]
P = scipy.spatial.ConvexHull(V)
scipy.spatial.convex_hull_plot_2d(P);
```



The obtained convex hull can be used to get the system of linear inequalities:

```
P.equations
```

```
array([[-0.        , -1.        ,  0.        ],
       [-1.        ,  0.        , -0.        ],
       [ 0.9486833 ,  0.31622777, -0.31622777],
       [ 0.31622777,  0.9486833 , -0.31622777]])
```

Note

The last two rows can be rescaled to give the original inequalities of (24.3).

24.4.2 Using SciPy to obtain the half space

The Scipy library has functionality to obtain the vertices from a given intersection of half spaces.

First we create the system of linear inequalities $Mx \leq b$, this is done passing: $(M, -b)$ as single matrix

```
halfspace = np.array(
    (
        (-1, 0, 0),
        (0, -1, 0),
        (3, 1, -1),
        (1, 3, -1),
    )
)
```

We need to pass a point known to be in the interior of the Polytope. Now we can obtain the vertices the original vertices from (24.1).

```
P = scipy.spatial.HalfspaceIntersection(halfspace, interior_point=np.array((1/5,
1/5)))
P.intersections
```

```
array([[2.77555756e-17, 3.33333333e-01],
       [2.50000000e-01, 2.50000000e-01],
       [0.00000000e+00, 0.00000000e+00],
       [3.33333333e-01, 2.77555756e-17]])
```

24.4.3 Carrying out row operations using NumPy

NumPy's array operations allow for straightforward row operations.

```
T_1 = np.array(
    (
        (1, 3, 1, 0, 1),
        (3, 1, 0, 1, 1),
    )
)
T_2 = T_1
T_2[0] = 3 * T_2[0] - T_2[1]
T_2
```

```
array([[ 0,  8,  3, -1,  2],
       [ 3,  1,  0,  1,  1]])
```

24.4.4 Pivoting tableaux with Nashpy

NashPy has some internal functionality for integer pivoting, which at present is not designed to be user facing but is nonetheless useable:

```
import nashpy as nash

T = nash.linalg.Tableau(T_2)
```

Having created this tableau we can get the indices of the basic and non-basic variables:

```
T.basic_variables
```

```
{0, 2}
```

```
T.non_basic_variables
```

```
{np.int64(1), np.int64(3)}
```

Note

Recall that Python uses 0 based indexing: the first variable/column corresponds to index 0.

We can pivot the tableau on a given column and given row:

```
T._pivot(column_index=1, pivot_row_index=1)
```

This pivoted on the second column (index 1) returning which non-basic variable becomes basic as a result.

We can look at the tableau:

```
T._tableau
```

```
array([[-24,   0,   3,  -9,  -6],
       [  3,   1,   0,   1,   1]])
```

24.5 Notable Research

The concept of pivoting and tableaux was first introduced by the mathematician George Dantzig in a 1947 report during his tenure at the Pentagon (Dantzig, 1947).

The initial publication of this idea is found in (Dantzig, 1951). Although Dantzig did not explicitly mention tableaux, they naturally emerged as an efficient method for traversing the edges of a polytope.

In (Dantzig, 2020) (originally published in 1990), Dantzig shares the following anecdote:

> "The first idea that would occur to anyone as a technique for solving a linear program, aside from the obvious one of moving through the interior of the convex set, is that of moving from one vertex to the next along the edges of the polyhedral set. I discarded this idea immediately as impractical in higher dimensional spaces. It seemed intuitively obvious that there would be far too many vertices and edges to wander over in the general case for such a method to be efficient.
>
> When Hurwicz came to visit me at the Pentagon in the summer of 1947, I told him how I had discarded this vertex-edge approach as intuitively inefficient for solving LP. I suggested instead that we study the problem in the geometry of columns rather than the usual one of the rows

– column geometry incidentally was the one I had used in my Ph.D. thesis on the Neyman-Pearson Lemma. We dubbed the new method 'climbing the bean pole.'"

A comprehensive historical overview is provided in (Todd, 2002), which includes several other notable publications.

Two significant works connect integer pivoting to game theory:

- In a 1951 book chapter, Dantzig established a connection for zero-sum games, as described in (Adler, 2013).
- (Lemke & Howson, 1964) introduces the Lemke-Howson algorithm, which employs integer pivoting to find Nash equilibria in general games.

24.6 Conclusion

Integer pivoting offers a powerful lens through which to understand movement across the vertices of polytopes, providing a concrete foundation for key algorithms in linear programming and game theory. Through the tableau representation, we are able to:

- Translate systems of inequalities into algebraic form;
- Perform exact row operations that correspond to movement along polytope edges;
- Interpret basic and non-basic variables as defining vertices and directions of movement.

These tools underpin the algorithms used for zero-sum games and Nash equilibrium computation.

Table 24.1 summarises the key concepts introduced in this appendix.

Concept	Description
Polytope (convex hull)	Set of convex combinations of finite points
Polytope (half-spaces)	Set of solutions to a system of linear inequalities
Slack variable	Converts an inequality into an equality with a non-negative remainder
Tableau	Augmented matrix form of a linear system with slack variables
Basic variable	Variable corresponding to a pivot column; used to determine a vertex
Non-basic variable	Set to 0 when solving for vertex from tableau
Pivoting	Row operations used to exchange basic and non-basic variables
Minimum ratio test	Selects the pivot row to preserve feasibility during a pivot
Integer pivoting	Pivoting using only integer row operations to maintain exact structure

Table A: Summary of integer pivoting

Important

Each tableau corresponds to a vertex of a polytope, and integer pivoting lets us move precisely from one vertex to a neighbouring one, forming the algorithmic core of many key results in game theory.

24.7 Solutions

24.7.1 Solution to Exercise: Polytope Representation

1. $V = \{(0,0), (0,1), (1,0), (1,1)\}$

This is the unit square. As the intersection of half-spaces:

$$\mathcal{P} = \{x \in \mathbb{R}^2 \text{ mid } x_1 \geq 0,\ x_2 \geq 0,\ x_1 \leq 1,\ x_2 \leq 1\} \tag{AL}$$

In standard form $Mx \leq b$:

$$M = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}. \tag{AM}$$

2. $V = \{(0,0), (0,1/4), (1,2/3), (2,1/5)\}$

We must find the convex hull of these four points. Computing the supporting half-spaces (using the convex hull):

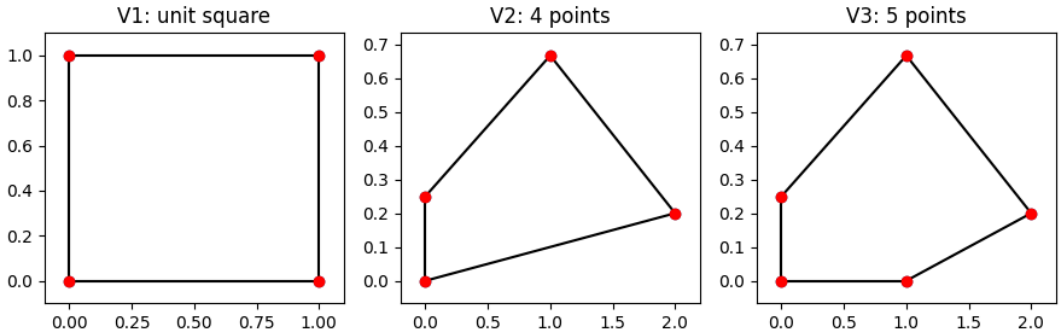
The edges of the convex hull connect:

- $(0,0)$ to $(0,1/4)$: left edge, $x_1 \geq 0$.
- $(0,0)$ to $(2,1/5)$: bottom edge. The line through these points has slope $1/10$, giving $x_2 \geq x_1/10$, i.e., $x_1 - 10x_2 \leq 0$.
- $(0,1/4)$ to $(1,2/3)$: upper-left edge. Slope $= (2/3 - 1/4)/1 = 5/12$. Line: $x_2 - 1/4 = (5/12)(x_1 - 0)$, i.e., $x_2 = 5x_1/12 + 1/4$, giving $-5x_1 + 12x_2 \leq 3$.
- $(1,2/3)$ to $(2,1/5)$: upper-right edge. Slope $= (1/5 - 2/3)/1 = -7/15$. Line: $x_2 - 2/3 = (-7/15)(x_1 - 1)$, i.e., $7x_1 + 15x_2 \leq 17$.
- $(2,1/5)$ to $(0,0)$: see bottom edge above (they share the same bounding line $x_1 - 10x_2 \leq 0$ together with $x_1 \leq 2$).

A complete half-space representation is:

$$x_1 \geq 0, \quad x_1 - 10x_2 \leq 0, \quad -5x_1 + 12x_2 \leq 3, \quad 7x_1 + 15x_2 \leq 17, \quad x_1 \leq 2. \tag{AN}$$

```
import scipy.spatial
import numpy as np
import matplotlib.pyplot as plt

V1 = np.array([[0,0],[0,1],[1,0],[1,1]], dtype=float)
hull1 = scipy.spatial.ConvexHull(V1)
fig, axes = plt.subplots(1, 3, figsize=(9, 3))

for ax, V, title in zip(
    axes,
    [
        np.array([[0,0],[0,1],[1,0],[1,1]], dtype=float),
        np.array([[0,0],[0,1/4],[1,2/3],[2,1/5]], dtype=float),
        np.array([[0,0],[0,1/4],[1,2/3],[2,1/5],[1,0]], dtype=float),
    ],
    ["V1: unit square", "V2: 4 points", "V3: 5 points"]
):
    hull = scipy.spatial.ConvexHull(V)
    scipy.spatial.convex_hull_plot_2d(hull, ax=ax)
    ax.scatter(V[:, 0], V[:, 1], color="red", zorder=5)
    ax.set_title(title)

plt.tight_layout()
```




```
# Show the half-space inequalities for V2
V2 = np.array([[0,0],[0,1/4],[1,2/3],[2,1/5]], dtype=float)
hull2 = scipy.spatial.ConvexHull(V2)
print("Half-space equations for V2 (each row: [a, b, c] means a*x1 + b*x2 + c <=
0):")
print(hull2.equations)
```

```
Half-space equations for V2 (each row: [a, b, c] means a*x1 + b*x2 + c <= 0):
[[ 0.09950372 -0.99503719 -0.        ]
 [ 0.42288547  0.90618314 -1.02700756]
 [-1.          0.         -0.        ]
 [-0.38461538  0.92307692 -0.23076923]]
```

3. $V = \{(0,0), (0,1/4), (1,2/3), (2,1/5), (1,0)\}$

The point $(1,0)$ lies inside or on the boundary of the convex hull of V from part 2 (it is below the bottom edge $x_1 - 10x_2 \leq 0$ at $x_1 = 1\colon 1 - 0 = 1 > 0$, so $(1,0)$ is actually outside that edge). Adding $(1,0)$ changes the convex hull.

```
V3 = np.array([[0,0],[0,1/4],[1,2/3],[2,1/5],[1,0]], dtype=float)
hull3 = scipy.spatial.ConvexHull(V3)
print("Half-space equations for V3:")
print(hull3.equations)
print("\nVertices of hull3:", V3[hull3.vertices])
```

```
Half-space equations for V3:
[[ 0.42288547  0.90618314 -1.02700756]
 [-1.          0.         -0.        ]
 [-0.38461538  0.92307692 -0.23076923]
 [-0.         -1.          0.        ]
 [ 0.19611614 -0.98058068 -0.19611614]]

Vertices of hull3: [[2.         0.2       ]
 [1.         0.66666667]
 [0.         0.25      ]
 [0.         0.        ]
 [1.         0.        ]]
```

24.7.2 Solution to Exercise: Basic and Non-Basic Variables

1. For

$$T^{(a)} = \begin{pmatrix} 1 & 0 & 5 & 1 & 1 \\ 0 & 1 & 4 & 9 & 1 \end{pmatrix} \tag{AO}$$

The columns (from left to right) correspond to variables x_1, x_2, x_3, x_4 (or s_1, s_2 if these are slack variables). A basic variable corresponds to a column that has exactly one nonzero entry (a unit vector). Inspecting the columns:

- Column 1 (x_1): $(1, 0)^T$, a unit vector. **Basic.**
- Column 2 (x_2): $(0, 1)^T$, a unit vector. **Basic.**
- Column 3 (x_3): $(5, 4)^T$, not a unit vector. **Non-basic.**
- Column 4 (x_4): $(1, 9)^T$, not a unit vector. **Non-basic.**

Basic variables: x_1, x_2. **Non-basic variables:** x_3, x_4.

Setting non-basic variables to zero: $x_3 = x_4 = 0$, the system gives $x_1 = 1$ and $x_2 = 1$, so the corresponding vertex is $(x_1, x_2) = (1, 1)$ (if x_3, x_4 are the original variables, otherwise the vertex is in the x_1, x_2 coordinates).

2. For

$$T^{(b)} = \begin{pmatrix} 4 & 8 & 1 & 0 & 1 \\ 8 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{AP}$$

- Column 1 (x_1): $(4, 8)^T$, not a unit vector. **Non-basic.**
- Column 2 (x_2): $(8, 1)^T$, not a unit vector. **Non-basic.**
- Column 3 (s_1): $(1, 0)^T$, a unit vector. **Basic.**
- Column 4 (s_2): $(0, 1)^T$, a unit vector. **Basic.**

Basic variables: s_1, s_2. **Non-basic variables:** x_1, x_2.

Setting non-basic variables to zero: $x_1 = x_2 = 0$, and from the last column, $s_1 = 1$, $s_2 = 1$. This corresponds to the origin vertex.

24.7.3 Solution to Exercise: Integer Pivoting

We perform integer pivoting on each non-basic variable for the two tableaux from the exercise above.

Pivoting on $T^{(a)}$: The non-basic variables are x_3 and x_4.

Pivot on x_3 (column 3):

The pivot column entries are $(5, 4)$. Both are positive, so we apply the minimum ratio test:

- Row 1: $b_1/T_{13} = 1/5$
- Row 2: $b_2/T_{23} = 1/4$

The minimum ratio is $1/5$ in row 1, so we pivot on row 1.

To eliminate x_3 from row 2 by integer pivoting (as defined in Definition: Integer Pivoting), multiply row 2 by 5 and subtract 4 times row 1, zeroing out column 3 in row 2:

$$\begin{aligned} \text{new row } 2 &= 5 \times (0, 1, 4, 9, 1) - 4 \times (1, 0, 5, 1, 1) \\ &= (0, 5, 20, 45, 5) - (4, 0, 20, 4, 4) = (-4, 5, 0, 41, 1). \end{aligned} \tag{AQ}$$

The updated tableau is:

$$T^{(a')} = \begin{pmatrix} 1 & 0 & 5 & 1 & 1 \\ -4 & 5 & 0 & 41 & 1 \end{pmatrix} \tag{AR}$$

The basic variables are now x_3 (from row 1) and x_2 (from row 2). Setting non-basic variables $x_1 = x_4 = 0$: from row 1, $5x_3 = 1 \Rightarrow x_3 = 1/5$; from row 2, $5x_2 = 1 \Rightarrow x_2 = 1/5$.

Pivot on x_4 (column 4) of $T^{(a)}$:

The pivot column entries are $(1, 9)$. Both positive. Minimum ratio test:

- Row 1: $1/1 = 1$
- Row 2: $1/9$

Minimum is $1/9$ in row 2. Pivot on row 2. Eliminate x_4 from row 1: multiply row 1 by 9 and subtract row 2 (integer pivoting):

$$\begin{aligned} \text{new row } 1 &= 9 \times (1, 0, 5, 1, 1) - 1 \times (0, 1, 4, 9, 1) \\ &= (9, 0, 45, 9, 9) - (0, 1, 4, 9, 1) = (9, -1, 41, 0, 8). \end{aligned} \tag{AS}$$

The updated tableau is:

$$T^{(a'')} = \begin{pmatrix} 9 & -1 & 41 & 0 & 8 \\ 0 & 1 & 4 & 9 & 1 \end{pmatrix} \tag{AT}$$

Pivoting on $T^{(b)}$: The non-basic variables are x_1 and x_2.

Pivot on x_1 (column 1):

The pivot column entries are $(4, 8)$. Both positive. Minimum ratio test:

- Row 1: $1/4$
- Row 2: $1/8$

Minimum is $1/8$ in row 2. Pivot on row 2. Eliminate x_1 from row 1:

$$\begin{aligned} \text{new row } 1 &= 8 \times (4, 8, 1, 0, 1) - 4 \times (8, 1, 0, 1, 1) \\ &= (32, 64, 8, 0, 8) - (32, 4, 0, 4, 4) = (0, 60, 8, -4, 4). \end{aligned} \tag{AU}$$

The updated tableau is:

$$T^{(b')} = \begin{pmatrix} 0 & 60 & 8 & -4 & 4 \\ 8 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{AV}$$

Basic variables are now x_1 (row 2) and s_1 (row 1). Setting $x_2 = s_2 = 0$: from row 2, $8x_1 = 1 \Rightarrow x_1 = 1/8$.

Pivot on x_2 (column 2):

Starting from $T^{(b)}$, the pivot column entries are $(8, 1)$. Both positive. Minimum ratio test:

- Row 1: $1/8$
- Row 2: $1/1 = 1$

Minimum is $1/8$ in row 1. Pivot on row 1. Eliminate x_2 from row 2:

$$\begin{aligned} \text{new row } 2 &= 8 \times (8, 1, 0, 1, 1) - 1 \times (4, 8, 1, 0, 1) \\ &= (64, 8, 0, 8, 8) - (4, 8, 1, 0, 1) = (60, 0, -1, 8, 7). \end{aligned} \tag{AW}$$

The updated tableau is:

$$T^{(b'')} = \begin{pmatrix} 4 & 8 & 1 & 0 & 1 \\ 60 & 0 & -1 & 8 & 7 \end{pmatrix} \tag{AX}$$

Basic variables are now s_1 (row 1) and x_2 (row 2). Setting $x_1 = s_2 = 0$: from row 1, $8x_2 + s_1 = 1$; from row 2, $8s_2 = 7$... Actually with $x_1 = s_2 = 0$: row 1 gives $8x_2 = 1 \Rightarrow x_2 = 1/8$.

```
import numpy as np

# Tableau T^(a)
T_a = np.array([[1, 0, 5, 1, 1],
                [0, 1, 4, 9, 1]], dtype=float)

# Pivot on column 3 (index 2), pivot row 1 (index 0): minimum ratio 1/5
T_a_pivot_x3 = T_a.copy()
pivot_col, pivot_row = 2, 0
# new_row2 = 5*row2 - 4*row1 (to zero column 3 in row 2)
T_a_pivot_x3[1] = T_a[1] * T_a[0, pivot_col] - T_a[0] * T_a[1, pivot_col]
print("T^(a) after pivot on x3 (col 3, row 1):")
print(T_a_pivot_x3)

# Pivot on column 4 (index 3) of T^(a), pivot row 2 (index 1): minimum ratio 1/9
T_a_pivot_x4 = T_a.copy()
pivot_col, pivot_row = 3, 1
# new_row1 = 9*row1 - row2 (to zero column 4 in row 1)
T_a_pivot_x4[0] = T_a[0] * T_a[1, pivot_col] - T_a[1] * T_a[0, pivot_col]
print("\nT^(a) after pivot on x4 (col 4, row 2):")
print(T_a_pivot_x4)
```

```
T^(a) after pivot on x3 (col 3, row 1):
[[ 1.  0.  5.  1.  1.]
 [-4.  5.  0. 41.  1.]]

T^(a) after pivot on x4 (col 4, row 2):
[[ 9. -1. 41.  0.  8.]
 [ 0.  1.  4.  9.  1.]]
```

```
# Tableau T^(b)
T_b = np.array([[4, 8, 1, 0, 1],
                [8, 1, 0, 1, 1]], dtype=float)

# Pivot on x1 (col 1, index 0), pivot row 2 (index 1): minimum ratio 1/8
T_b_pivot_x1 = T_b.copy()
pivot_col, pivot_row = 0, 1
T_b_pivot_x1[0] = T_b[0] * T_b[1, pivot_col] - T_b[1] * T_b[0, pivot_col]
print("T^(b) after pivot on x1 (col 1, row 2):")
print(T_b_pivot_x1)

# Pivot on x2 (col 2, index 1), pivot row 1 (index 0): minimum ratio 1/8
T_b_pivot_x2 = T_b.copy()
pivot_col, pivot_row = 1, 0
T_b_pivot_x2[1] = T_b[1] * T_b[0, pivot_col] - T_b[0] * T_b[1, pivot_col]
print("\nT^(b) after pivot on x2 (col 2, row 1):")
print(T_b_pivot_x2)
```

```
T^(b) after pivot on x1 (col 1, row 2):
[[ 0. 60.  8. -4.  4.]
 [ 8.  1.  0.  1.  1.]]
```

```
T^(b) after pivot on x2 (col 2, row 1):
[[ 4.  8.  1.  0.  1.]
 [60.  0. -1.  8.  7.]]
```

25. About the Authors

25.1 Vincent Knight

Vincent Knight is a Chair in the School of Mathematics at Cardiff University, where he works on game theory, applied mathematics, and open-source scientific software. His research spans the iterated prisoner's dilemma, the modelling of emergency healthcare systems, and the reproducible practice of computational science. He is the author and maintainer of several open-source libraries, including Nashpy and the Axelrod-Python project. He wrote the book *Python for Mathematics* and, with Dr Geraint Palmer, *Applied Mathematics with Open-Source Software: Operational Research Problems with Python and R* (Routledge). More of his work can be found at vknight.org.

25.2 James Brown

The illustrations in this book are the work of James Brown. James is an artist, an award-winning filmmaker, and a game developer whose work is drawn to group interactions and systems: connection, trust, and the conditions under which people cooperate. He studied Intermedia at the Elam School of Fine Arts and spent several years as an independent game developer before turning to the games, simulations, and digital art that he now makes.

Much of that work lives at his wonderful site, nonzerosum.games, which he describes as an exploration of non-zero-sum games and of the idea that we are more than the sum of our parts. It is well worth a visit, and we are delighted that his art sits alongside the mathematics in these pages.

References

Accinelli, E., & Carrera, E. J. S. (2011). Evolutionarily stable strategies and replicator dynamics in asymmetric two-population games. In *Dynamics, Games and Science I: DYNA 2008, in Honor of Maurıcio Peixoto and David Rand, University of Minho, Braga, Portugal, September 8-12, 2008* (pp. 25–35). Springer.

Adler, I. (2013). The equivalence of linear programs and zero-sum games. *International Journal of Game Theory*, *42*, 165–177.

Antal, T., & Scheuring, I. (2006). Fixation probability and fixation time in the Moran process with two types of individuals. *Bulletin of Mathematical Biology*, *68*(8), 1923–1944.

Arrow, K. J. (1951). *Social Choice and Individual Values*. Wiley.

Axelrod, R. (1980). Effective choice in the prisoner's dilemma. *Journal of Conflict Resolution*, *24*(1), 3–25.

Axelrod, R. (1980). More effective choice in the prisoner's dilemma. *Journal of Conflict Resolution*, *24*(3), 379–403.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books.

Basanta, D., Hatzikirou, H., & Deutsch, A. (2008). Studying the emergence of invasiveness in tumours using game theory. *The European Physical Journal B*, *63*, 393–397.

Bianchini, M., Gori, M., & Scarselli, F. (2005). Inside PageRank. *ACM Transactions on Internet Technology*, *5*(1), 92–128. https://doi.org/10.1145/1052934.1052938

Bondareva, O. N. (1963). Some applications of linear programming methods to the theory of cooperative games. *Problemy Kibernetiki*, *10*, 119–139.

Borda, J.-C. de. (1781). Mémoire sur les élections au scrutin. *Histoire De L'académie Royale Des Sciences*.

Boyd, N. T., Gabriel, S. A., Rest, G., & Dumm, T. (2023). Generalized Nash equilibrium models for asymmetric, non-cooperative games on line graphs: Application to water resource systems. *Computers & Operations Research*, *154*, 106194.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Braess, D. (1968).¨Uber ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, *12*, 258–268.

Bredereck, R., Faliszewski, P., Igarashi, A., Lackner, M., & Skowron, P. (2018). Multiwinner elections with diversity constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Caritat, M. de C. Marie Jean Antoine Nicolas de. (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie Royale.

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, *36*(5), 1726–1730.

Chiappori, P.-A., Levitt, S., & Groseclose, T. (2002). Testing mixed-strategy equilibria when players are heterogeneous: The case of penalty kicks in soccer. *American Economic Review*, *92*(4), 1138–1151.

Clutton-Brock, T. H., Guinness, F. E., & Albon, S. D. (1982). *Red Deer: Behavior and Ecology of Two Sexes*. University of Chicago Press.

Conitzer, V., & Sandholm, T. (2002). Complexity of manipulating elections with few candidates. *AAAI/IAAI*, 314–319.

Couto, M. C., Giaimo, S., & Hilbe, C. (2022). Introspection dynamics: a simple model of counterfactual learning in asymmetric games. *New Journal of Physics*, *24*(6), 63033.

Dantzig, G. B. (1951). Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, *13*, 339–347.

Dantzig, G. B. (2020). Impact of linear programming on computer development. In *Computers in Mathematics* (pp. 233–240). CRC Press.

Dantzig, G. B. (1947). *Maximization of a Linear Function of Variables Subject to Linear Inequalities* (Techreport No. P-38).

Daskalakis, C., Goldberg, P. W., & Papadimitriou, C. H. (2009). The complexity of computing a Nash equilibrium. *Communications of the ACM*, *52*(2), 89–97.

Deng, X., & Papadimitriou, C. H. (1994). On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, *19*(2), 257–266.

Dresher, M. (1970). Probability of a pure equilibrium point in n-person games. *Journal of Combinatorial Theory*, *8*(1), 134–145.

Elkind, E., Faliszewski, P., Skowron, P., & Slinko, A. (2017). Properties of multiwinner voting rules. *Social Choice and Welfare*, *48*, 599–632.

Enquist, M., & Leimar, O. (1987). Evolution of fighting behaviour: the effect of variation in resource value. *Journal of Theoretical Biology*, *127*(2), 187–205. https://doi.org/10.1016/S0022-5193(87)80130-3

Euler, L. (1768). *Institutionum Calculi Integralis*. Academiae Imperialis Scientiarum Petropolitanae.

Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. A. (2010). Using complexity to protect elections. *Communications of the ACM*, *53*(11), 74–82.

Fisher, R. A. (1930). *The Genetical Theory of Natural Selection*. Clarendon Press.

Friedman, J. W. (1971). A non-cooperative equilibrium for supergames. *The Review of Economic Studies*, *38*(1), 1–12.

Friedman, J. W. (1973). A non-cooperative equilibrium for supergames: A correction. *The Review of Economic Studies*, *40*(3), 435.

Fudenberg, D., & Maskin, E. (1986). The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, *54*(3), 533–554.

Gale, D., & Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, *69*(1), 9–15.

Gibbard, A. (1973). Manipulation of voting schemes: a general result. *Econometrica: Journal of the Econometric Society*, 587–601.

Gilboa, I., & Matsui, A. (1991). Social stability and equilibrium. *Econometrica*, *59*(3), 859–867.

Gillies, D. B. (1959). Solutions to general non-zero-sum games. In A. W. Tucker & R. D. Luce (Eds.), *Contributions to the Theory of Games, Volume IV* (Issue 40, pp. 47–85). Princeton University Press.

Glynatsi, N. E., Knight, V., & Harper, M. (2024). Properties of winning Iterated Prisoner's Dilemma strategies. *PLOS Computational Biology*, *20*(12), e1012644.

Glynatsi, N. E., Knight, V., & Lee, T. E. (2018). An evolutionary game theoretic model of rhino horn devaluation. *Ecological Modelling*, *389*, 33–40.

Grossman, S. J., & Perry, M. (1986). Perfect sequential equilibrium. *Journal of Economic Theory*, *39*(1), 97–119.

Gusfield, D., & Irving, R. W. (1989). *The stable marriage problem: structure and algorithms*. MIT press.

Hairer, E., & Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems* (2nd ed., Vol. 14, Pt. 14, Chapter 14). Springer.

Hairer, E., Nrsett, S. P., & Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems* (2nd ed., Vol. 8, Pt. 8, Chapter 8). Springer.

Haldane, J. B. S. (1927). A Mathematical Theory of Natural and Artificial Selection, Part IV. *Mathematical Proceedings of the Cambridge Philosophical Society*, *23*(5), 838–844.

Haldane, J. B. S. (1932). *The Causes of Evolution*. Longmans, Green.

Hamilton, W. D. (1967). Extraordinary sex ratios. *Science*, *156*(3774), 477–488. https://doi.org/10.1126/science.156.3774.477

Harper, M. (2019). python-ternary: Ternary Plots in Python. *Zenodo 10.5281/zenodo.594435*. https://doi.org/10.5281/zenodo.594435

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Harsanyi, J. C. (1973). Oddness of the number of equilibrium points: a new proof. *International Journal of Game Theory*, *2*(1), 235–250.

Harshman, L. G., & Zera, A. J. (2007). The cost of reproduction: the devil in the details. *Trends in Ecology & Evolution*, *22*(2), 80–86. https://doi.org/10.1016/j.tree.2006.10.008

Hilbe, C., Nowak, M. A., & Sigmund, K. (2013). Evolution of extortion in iterated prisoner's dilemma games. *Proceedings of the National Academy of Sciences*, *110*(17), 6913–6918.

Hofbauer, J., & Sigmund, K. (1998). *Evolutionary Games and Population Dynamics*. Cambridge University Press.

Holliday, W. H., & Pacuit, E. (2025). pref_voting: The Preferential Voting Tools package for Python. *Journal of Open Source Software*, *10*(105), 7020. https://doi.org/10.21105/joss.07020

Jin, D., Sergeeva, E., Weng, W.-H., Chauhan, G., & Szolovits, P. (2022). Explainable deep learning in healthcare: A methodological survey from an attribution view. *Wires Mechanisms of Disease*, *14*(3), e1548.

John, F. (1948,). *Extremum Problems with Inequalities as Side Conditions* [Technical Report].

Jordan, J. D., Melouk, S. H., & Perry, M. B. (2009). Optimizing football game play calling. *Journal of Quantitative Analysis in Sports*, *5*(2).

Kakutani, S. (1941). A generalization of Brouwer's fixed point theorem. *Duke Mathematical Journal*, *8*(3), 457–459.

Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. Of Mathematics, Univ. Of Chicago*.

Kemeny, J. G., & Snell, J. L. (1976). *Finite Markov chains*. Springer-Verlag.

Knight, V. A., & Harper, P. R. (2013). Selfish routing in public services. *European Journal of Operational Research*, *230*(1), 122–132.

Knight, V. A., Campbell, O., Harper, M., Langner, K. M., Campbell, J., Campbell, T., Carney, A., Chorley, M., Davidson-Pilon, C., Glass, K., & others. (2016). An open framework for the reproducible study of the iterated prisoner's dilemma. *Journal of Open Research Software*, *4*(1).

Knight, V., & Campbell, J. (2018). Nashpy: A Python library for the computation of Nash equilibria. *Journal of Open Source Software*, *3*(30), 904.

Knight, V., Harper, M., Glynatsi, N. E., & Campbell, O. (2018). Evolution reinforces cooperation with the emergence of self-recognition mechanisms: An empirical study of strategies in the Moran process for the iterated prisoner's dilemma. *Plos One*, *13*(10), e0204981.

Knight, V., Harper, M., Glynatsi, N. E., & Gillard, J. (2024). Recognising and evaluating the effectiveness of extortion in the Iterated Prisoner's Dilemma. *Plos One*, *19*(7), e0304641.

Knight, V., Komenda, I., & Griffiths, J. (2017). Measuring the price of anarchy in critical care unit interactions. *Journal of the Operational Research Society*, *68*(6), 630–642.

Knuth, D. E. (1976). *Mariages stables et leurs relations avec d'autres problèmes combinatoires*. Les Presses de l'Université de Montréal.

Komarova, N. L., Niyogi, P., & Nowak, M. A. (2001). The evolutionary dynamics of grammar acquisition. *Journal of Theoretical Biology*, *209*(1), 43–59.

Kreps, D. M., & Wilson, R. (1982). Sequential equilibria. *Econometrica: Journal of the Econometric Society*, 863–894.

Kruuk, L. E. B., Slate, J., Pemberton, J. M., Brotherstone, S., Guinness, F., & Clutton-Brock, T. (2002). Antler size in red deer: heritability and selection but no evolution. *Evolution*, *56*(8), 1683–1695. https://doi.org/10.1111/j.0014-3820.2002.tb01480.x

Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear Programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 481–492.

Kutta, W. (1901). Beiträge zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift Für Mathematik Und Physik*, *46*, 435–453.

Kwak, J., Lees, M. H., Cai, W., & Ong, M. E. (2020). Modeling helping behavior in emergency evacuations using volunteer's dilemma game. *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part I 20*, 513–523.

Langville, A., & Meyer, C. (2004). Deeper Inside PageRank. *Internet Mathematics*, *1*(3), 335–380. https://doi.org/10.1080/15427951.2004.10129091

Lee, T. E., & Roberts, D. L. (2016). Devaluing rhino horns as a theoretical game. *Ecological Modelling*, *337*, 73–78.

Lemke, C. E., & Howson, J. T., Jr. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, *12*(2), 413–423.

Lieberman, E., Hauert, C., & Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature*, *433*(7023), 312–316.

Lv, S., Li, J., & Zhao, C. (2023). The evolution of cooperation in voluntary public goods game with shared-punishment. *Chaos, Solitons & Fractals*, *172*, 113552.

Maschler, M., Zamir, S., & Solan, E. (2020). *Game theory*. Cambridge University Press.

Maynard Smith, J. (1982). *Evolution and the Theory of Games*. Cambridge University Press.

Maynard Smith, J., & Price, G. R. (1973). The Logic of Animal Conflict. *Nature*, *246*, 15–18. https://doi.org/10.1038/246015a0

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: symbolic computing in Python. *Peerj Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Milgrom, P. R., & Weber, R. J. (1982). A theory of auctions and competitive bidding. *Econometrica: Journal of the Econometric Society*, 1089–1122.

Milgrom, P., & Roberts, J. (1982). Predation, reputation, and entry deterrence. *Journal of Economic Theory*, *27*(2), 280–312.

Moen, R. A., Pastor, J., & Cohen, Y. (1999). Antler growth and extinction of Irish elk. *Evolutionary Ecology Research*, *1*(2), 235–249. http://www.evolutionary-ecology.com/abstracts/v01/1026.html

Moran, P. A. P. (1958). Random processes in genetics. *Mathematical Proceedings of the Cambridge Philosophical Society*, *54*(1), 60–71.

Myerson, R. B. (1981). Optimal auction design. *Mathematics of Operations Research*, *6*(1), 58–73.

Nash Jr, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, *36*(1), 48–49.

Neumann. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, *100*(1), 295–320.

Neumann. (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, *100*(1), 295–320.

Neumann, J. von, & Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Nowak, M. A. (2006). *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press.

Nowak, M. A., & Sigmund, K. (2004). Evolutionary dynamics of biological games. *Science*, *303*(5659), 793–799.

O'Mahony, S. (2007). The governance of open source initiatives: what does it mean to be community managed?. *Journal of Management & Governance*, *11*, 139–150.

Ohtsuki, H., Pacheco, J. M., & Nowak, M. A. (2007). Evolutionary Graph Theory: Breaking the Symmetry between Interaction and Replacement. *Journal of Theoretical Biology*, *246*(4), 681–694. https://doi.org/10.1016/j.jtbi.2007.01.024

Osborne, J. A. (2003). Markov Chains for the \textit{RISK} Board Game Revisited. *Mathematics Magazine*, *76*(2), 129–135. https://doi.org/10.1080/0025570X.2003.11953165

Osborne, M. J., & others. (2004). *An introduction to game theory* (Vol. 3, Issue 3). Springer.

Pacuit, E. (2019). Voting methods. *Stanford Encyclopedia of Philosophy*.

Palm, G. (1984). Evolutionary stable strategies and game dynamics for n-person games. *Journal of Mathematical Biology*, *19*, 329–334.

Palmer, G. I., Harper, P. R., & Knight, V. A. (2018). Modelling deadlock in open restricted queueing networks. *European Journal of Operational Research*, *266*(2), 609–621. https://doi.org/10.1016/j.ejor.2017.10.039

Panayides, M., Knight, V., & Harper, P. (2023). A game theoretic model of the behavioural gaming that takes place at the EMS-ED interface. *European Journal of Operational Research*, *305*(3), 1236–1258.

Petzold, L. R. (1983). Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, *4*(1), 136–148.

Press, W. H., & Dyson, F. J. (2012). Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, *109*(26), 10409–10413.

Price, G. R. (1970). Selection and covariance. *Nature*, *227*, 520–521. https://doi.org/10.1038/227520a0

Rosenthal, R. W. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, *2*(1), 65–67.

Roth, A. E. (1984). The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory. *Journal of Political Economy*, *92*(6), 991–1016.

Roth, A. E., Sönmez, T., & Ünver, M. U. (2004). Kidney Exchange. *Quarterly Journal of Economics*, *119*(2), 457–488.

Roughgarden, T. (2010). Algorithmic game theory. *Communications of the ACM*, *53*(7), 78–86.

Roughgarden, T., & Tardos, É. (2002). How bad is selfish routing?. *Journal of the ACM (JACM)*, *49*(2), 236–259.

Roughgarden, T. A. (2002). *Selfish routing*. Cornell University.

Runge, C. (1895).¨Uber die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, *46*(2), 167–178.

Satterthwaite, M. A. (1975). Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, *10*(2), 187–217.

Savani, R., & Turocy, T. L. (2024,). *Gambit: The package for computation in game theory, Version 16.2. 0*.

Savani, R., & Von Stengel, B. (2004). Exponentially many steps for finding a Nash equilibrium in a bimatrix game. *45th Annual IEEE Symposium on Foundations of Computer Science*, 258–267.

Schmeidler, D. (1969). The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, *17*(6), 1163–1170. https://doi.org/10.1137/0117107

Selten, R. (1965). Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit: Teil i: Bestimmung des dynamischen preisgleichgewichts. *Zeitschrift Für Die Gesamte Staatswissenschaft/journal of Institutional and Theoretical Economics*, (H. 2), 301–324.

Selten, R. (1978). The chain store paradox. *Theory and Decision*, *9*(2), 127–159.

Selten, R., & Bielefeld, R. S. (1988). *Reexamination of the perfectness concept for equilibrium points in extensive games*. Springer.

Shapley, L. S., & others. (1953). *A value for n-person games*.

Shapley, L. S. (1967). On balanced sets and cores. *Naval Research Logistics Quarterly*, *14*(4), 453–460. https://doi.org/10.1002/nav.3800140404

Shapley, L. S. (1971). Cores of convex games. *International Journal of Game Theory*, *1*(1), 11–26. https://doi.org/10.1007/BF01753431

Snell, J. L. (1959). Finite Markov Chains and their Applications. *The American Mathematical Monthly*, *66*(2), 99–104. https://doi.org/10.1080/00029890.1959.11989252

Steinberg, R., & Zangwill, W. I. (1983). The prevalence of Braess' paradox. *Transportation Science*, *17*(3), 301–318.

Strumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, *11*, 1–18.

Tan, B. (1997). Markov Chains and the \textit{RISK} Board Game. *Mathematics Magazine*, *70*(5), 349–357. https://doi.org/10.1080/0025570X.1997.11996573

Taylor, P. D., & Jonker, L. B. (1978). Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, *40*(1–2), 145–156.

Tedeschi, M. N., Hose, T. M., Mehlman, E. K., Franklin, S., & Wong, T. E. (2023). Improving models for student retention and graduation using Markov chains. *PLOS ONE*, *18*(6), e0287775. https://doi.org/10.1371/journal.pone.0287775

Todd, M. J. (2002). The many facets of linear programming. *Mathematical Programming*, *91*(3), 417–436.

Traulsen, A., & Nowak, M. A. (2006). Evolution of cooperation by kin selection and group selection in finite populations. *Proceedings of the National Academy of Sciences*, *103*(29), 10952–10955.

Traulsen, A., Claussen, J. C., & Hauert, C. (2005). Coevolutionary dynamics: from finite to infinite populations. *Physical Review Letters*, *95*(23), 238701.

Traulsen, A., Nowak, M. A., & Pacheco, J. M. (2006). Pairwise comparison and selection temperature in evolutionary game dynamics. *Journal of Theoretical Biology*, *246*(3), 522–529.

Vanderbei, R. J. (2010). *Linear Programming: Foundations and Extensions* (Softcover reprint of hardcover 3rd Edition 2008). Springer Science+Business Media.

Varian, H. R. (2007). Position auctions. *International Journal of Industrial Organization*, *25*(6), 1163–1178.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, *16*(1), 8–37.

Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, *1*(5), 767–768.

Watson, J. (2002). Strategy: an introduction to game theory. *(No Title)*.

Webb, J. N. (2007). *Game theory: decisions, interaction and Evolution*. Springer Science & Business Media.

Weitz, J. S., Eksin, C., Paarporn, K., Brown, S. P., & Ratcliff, W. C. (2016). An oscillating tragedy of the commons in replicator dynamics with game-environment feedback. *Proceedings of the National Academy of Sciences*, *113*(47), E7518–E7525.

Wiese, S. C., & Heinrich, T. (2022). The frequency of convergent games under best-response dynamics. *Dynamic Games and Applications*, 1–12.

Wilde, H., Knight, V., & Gillard, J. (2020). Matching: A python library for solving matching games. *Journal of Open Source Software*, *5*(48), 2169.

Wilson, R. (1971). Computing equilibria of n-person games. *SIAM Journal on Applied Mathematics*, *21*(1), 80–87.

Wilson, R. B. (1967). Competitive bidding with asymmetric information. *Management Science*, *13*(11), 816–820.

Wright, S. (1931). Evolution in Mendelian populations. *Genetics*, *16*(2), 97–159.

Youn, H., Gastner, M. T., & Jeong, H. (2008). Price of anarchy in transportation networks: efficiency and optimality control. *Physical Review Letters*, *101*(12), 128701.

Young, H. P. (1993). The evolution of conventions. *Econometrica: Journal of the Econometric Society*, 57–84.