

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Фізико-математичний факультет

Кафедра математичного аналізу та теорії ймовірності

«На правах рукопису»

УДК 519.863

До захисту допущено:

Завідувач кафедри

_____ О.І. Клесов

«__» _____ 20__ р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Страхова та фінансова
математика»**

зі спеціальності 111 «Математика»

**на тему: «Застосування робастної оптимізації до ігрової моделі
ринкової конкуренції»**

Виконала:

студентка II курсу магістратури, групи ОМ-41мп

Сатир Яна Миколаївна _____

Науковий керівник:

кандидат фізико-математичних наук, доцент

Алексєєва Ірина Віталіївна _____

Рецензент:

професор кафедри телекомунікацій,

доктор технічних наук, професор,

Лисенко Олександр Іванович _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-математичний факультет
Кафедра математичного аналізу та теорії ймовірності

Рівень вищої освіти – другий (магістерський)

Спеціальність – 111 «Математика»

Освітньо-професійна програма «Страхова та фінансова математика»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.І. Клесов

«__» _____ 20__ р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Сатир Яні Миколаївні

1. Тема дисертації «Застосування робастної оптимізації до ігрової моделі ринкової конкуренції», науковий керівник дисертації Алексеева Ірина Віталіївна, кандидат фізико-математичних наук, доцент, затверджені наказом по університету від «06» листопада 2025 р. № 4843-с
2. Термін подання студентом дисертації 18.12.2025
3. Об'єкт дослідження: модель ринкової конкуренції, як біматрична гра.
4. Вихідні дані: матриці прибутків для розв'язання математичної моделі, методи розв'язання лінійної задачі робастної оптимізації.
5. Перелік завдань, які потрібно розробити:
 - побудувати математичну модель ринкової конкуренції у вигляді біматричної гри
 - знайти гарантованого прибуток для кожного гравця

- визначити захисні стратегії гравців
- розв'язати математичну модель у вигляді робастної оптимізаційної задачі
- проаналізувати та порівняти отримані результати

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 21 слайд.

7. Дата видачі завдання 03.09.2025

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Формулювання теми дипломної роботи, розробка плану.	03.09.2025–04.09.2025	Виконала
2	Пошук та аналіз літератури на тему дипломної роботи.	05.09.2025–17.09.2025	Виконала
3	Виконання практичної частини. Аналіз отриманих даних.	18.09.2025–21.10.2025	Виконала
4	Написанні попереднього тексту дисертації.	22.10.2025–14.11.2025	Виконала
5	Редагування тексту дисертації. Формулювання висновків.	15.11.2025–08.12.2025	Виконала
6	Підготовка презентації.	09.12.2025–17.12.2025	Виконала

Студент

Яна САТИР

Науковий керівник

Ірина АЛЕКСЄЄВА

РЕФЕРАТ

Магістерська дисертація: 55 сторінки, 6 рисунків,
8 таблиць, 4 додатків, 10 першоджерел.

В даній магістерській дисертації досліджено застосування методів робастної оптимізації для ігрової моделі ринкової конкуренції, які передбачають побудову робастного аналога детермінованої задачі. У роботі розглянуто прямокутну та еліпсоїдальну області невизначеності. Для кожного типу невизначеності побудовані робастні оптимізаційні задачі. Проведено числовий експеримент, що демонструє змінення оптимальних змішаних стратегій при різних параметрах множини невизначеності. Отримані результати проаналізовані та порівняні.

Мета роботи: побудова математичної моделі ринкової конкуренції у вигляді біматричної гри, визначення гарантованого результату для кожного гравця, знаходження захисних стратегій, розв'язання гри у вигляді робастної оптимізаційної задачі.

Ключові слова: матрична гра, біматрична гра, робастна оптимізація, ринкова конкуренція, невизначеність.

ABSTRACT

Master`s thesis: 55 pages, 6 picture,
8 tables, 4 applications, 10 primary sources.

This master's thesis investigates the application of robust optimization methods to a game model of market competition, which involves constructing a robust analogue of a deterministic problem. The work considers rectangular and ellipsoidal regions of uncertainty. Robust optimization problems were constructed for each type of uncertainty. A numerical experiment was conducted to demonstrate changes in optimal mixed strategies for different parameters of the uncertainty set. The results obtained were analyzed and compared.

Purpose of the work: to construct a mathematical model of market competition in the form of a two-matrix game, to find a guaranteed result for each player, to find defensive strategies, to solve the game in the form of a robust optimization problem.

Keywords: matrix game, bimatrix game, robust optimization, market competition, uncertainty.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. Теорія ігор.....	10
1.1 Матрична гра в чистих стратегіях	10
1.2 Матрична гра в змішаних стратегіях.....	11
1.3 Біматрична гра.....	12
РОЗДІЛ 2. Робастна оптимізація	14
2.1 Поняття про робастну оптимізацію.....	14
РОЗДІЛ 3. Области невизначеності	18
3.1 Прямокутна область невизначеності (Box, Soyster, 1973)	18
3.2 Еліпсоїдальна область невизначеності (Ellipsoidal set, Ben-Tal & Nemirovski, 1998).....	19
3.3 Перетин прямокутної та еліпсоїдальної області невизначеності.....	20
3.4 Невизначеність в лівих частинах обмежень (LHS).....	21
РОЗДІЛ 4. Перевірка методу.....	25
РОЗДІЛ 5. Оптимізація ігрової моделі ринкової конкуренції.....	31
5.1 Постановка завдання дослідження	31
5.2 Результати обчислень	36
Висновки	41
Джерела	42
Додаток 1. Гарантований результат та захисні стратегії для біматричної гри змішаних стратегій.	43
Додаток 2. Розв’язання робастної оптимізації для прямокутної області невизначеності.....	45

Додаток 3. Розв'язання робастної оптимізації для еліпсоїдальної області невизначеності..... 48

Додаток 4. Обчислення перетину прямокутної та еліпсоїдальної невизначеності робастної оптимізації біматричної гри. 51

ВСТУП

Сучасні ринкові системи функціонують за умови невизначеності через обмежену інформацію, мінливість економічного середовища та впливу зовнішніх чинників. В таких умовах використання класичної детермінованої оптимізаційної моделі, що базується на точно заданих параметрах, не завжди забезпечує стабільні результати.

Для аналізу стратегічної поведінки гравців використовують теорію ігор. Теорія ігор – це математична дисципліна, яка вивчає закономірності вибору стратегій за умови, що результат для кожного гравця залежить не лише від його власних дій, а й від стратегій іншого гравця.

Ключовою роботою в становленні теорії ігор як окремого напрямку стала публікація в 1944 році книги «Теорія ігор і економічна поведінка» Джона фон Неймана і економіста Оскара Моргенштерна, в якій вперше теорія ігор була застосована для вирішення бізнес задач з неповною інформацією. Подальший розвиток зумовлений роботами Джона Неша, який увів поняття рівноваги, що стало ключовим інструментом аналізу ігор з незалежним вибором стратегій. Згодом теорія ігор набула широкого використання в економіці, фінансах, політичних науках, біології та інших галузях.

В економічних дослідженнях особливу увагу приділяють ігровим моделям ринкової конкуренції. Проте класичні ігрові моделі зазвичай базуються на припущеннях, тому розв'язок може втратити надійність отриманих результатів.

Одним із підходів врахування невизначеностей в задачах, які дозволяють будувати рішення, що залишаються ефективними навіть при найгірших реалізаціях параметрів в межах визначених множин — є робастна оптимізація.

A.L.Soyster [1] в 1973 році вперше запропонував такий підхід. Хоча перші дослідження відбувалися в 70-х роках, сам напрям робастної оптимізації формувався протягом останніх 15 років та продовжує активно розвиватись. В науковій літературі представлена значна кількість робіт, зокрема роботи A.Ven-

Tal, A. Nemirovski [2], на яких дана дисертація буде ґрунтуватись. Застосуванню робастної оптимізації до теорії ігор присвячені роботи M. Aghassi, D. Bertsimas [3], G. P. Crespi, D. Radi, M. Rocca [5].

Поєднання методів теорії ігор та робастної оптимізації дає змогу будувати математичні моделі ринкової конкуренції та визначати стратегії, що будуть стійкими до невизначеностей.

Мета роботи:

1. Побудувати математичну модель ринкової конкуренції у вигляді біматричної гри.

2. Знайти захисні стратегії і гарантований прибуток кожного гравця як у детермінованій задачі, так і в робастних аналогах із різною геометрією областей невизначеності.

3. Проаналізувати вплив різних типів невизначеності на оптимальні стратегії та прибутковість гравців, а також оцінити ефективність застосування робастної оптимізації для формування цінових стратегій в умовах ризику та нестабільності ринку.

РОЗДІЛ 1. Теорія ігор

1.1 Матрична гра в чистих стратегіях

Означення 1.1 [7]

Грою в стратегічній (нормальній) формі називається трійка

$$G = \left\langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \right\rangle,$$

де $N = \{1, 2, \dots, n\}$ – скінченна множина гравців; S_i – множина стратегій i -го гравця, $i \in N$, $S = S_1 \times S_2 \times \dots \times S_n$; $u_i : S \rightarrow \mathbb{R}$ – функція, яка співставляє вектору стратегій $s = (s_i)_{i \in N} \in S$ виграш $u_i(s)$ i -го гравця $\forall i \in N$.

Матрична гра — це модель взаємодії двох гравців з нульовою сумою, де кожен має скінченну множину стратегій. Тобто,

$$u_1(s_1, s_2) + u_2(s_1, s_2) = 0.$$

В матричній грі виграш одного гравця дорівнює втраті іншого, тому результати гри можна представити у вигляді однієї матриці, в якій рядки відповідають стратегіям першого гравця, стовпці – стратегіям другого, а клітини на перетині рядків і стовпців відповідають ситуаціям в грі (профілю стратегій). Нехай задана матриця

$$A = (a_{ij})_{m \times n},$$

де $a_{ij} > 0, i \in I = \{1, \dots, m\}, j \in J = \{1, \dots, n\}$. Елемент a_{ij} виграш гравця A та програш гравця B , якщо гравець A обирає стратегію A_i , а гравець B відповідно стратегію B_j , які називаються чистими стратегіями гравців.

Вважається, що кожен гравець обирає для себе найкращу стратегію, тобто перший гравець намагаються максимізувати свій виграш, а другий гравець мінімізувати свій програш. Позначимо:

$$\underline{v} = \max_i \min_j a_{ij},$$

$$\bar{v} = \min_j \max_i a_{ij}.$$

Величину \underline{v} називають нижньою ціною гри, або максиміном, а \bar{v} – верхньою ціною гри або мінімаксом, причому $\underline{v} \leq \bar{v}$. Ці значення характеризують найгірший та найкращий можливий результат для кожного гравця у грі.

Якщо $\underline{v} = \bar{v}$, то $v = \underline{v} = \bar{v}$, де v – ціна гри:

$$v = \max_i \min_j a_{ij} = \min_j \max_i a_{ij}.$$

Означення 1.2

В матричній грі ситуація (i^*, j^*) називається ситуацією рівноваги або сідловою точкою, якщо

$$a_{ij^*} \leq a_{i^*j^*} \leq a_{i^*j}, \forall i = \overline{1, m}, j = \overline{1, n}.$$

Тобто, сідлова точка – це така комбінація стратегій, при якій жоден гравець не може покращити свій результат, змінивши стратегію одноосібно. Якщо така точка існує, то вона визначає оптимальні чисті стратегії для обох гравців (i^*, j^*) і ціну гри $v = a_{i^*j^*}$.

Таким чином, якщо нижня та верхня ціна гри співпадають, то гра має сідлову точку, а значення v є оптимальним для обох гравців.

1.2 Матрична гра в змішаних стратегіях

Якщо матрична гра немає сідлової точки, то розглядають змішане розширення гри. Це дозволяє знайти рівновагу навіть у складних випадках, коли жодна чиста стратегія не є оптимальною.

Змішана стратегія гравця – це ймовірнісний розподіл на множині чистих стратегій гравця.

В змішаних стратегіях перший гравець обирає стратегію

$$x \in \Delta_m = \left\{ x \geq 0, \sum_{i=1}^m x_i = 1 \right\}, \text{ а другий – стратегію } y \in \Delta_n = \left\{ y \geq 0, \sum_{j=1}^n y_j = 1 \right\}.$$

Теорема 1.1 (Дж. Фон Нейман (1928)) [7]

Будь-яка гра з нульовою сумою в якій кожен гравець має скінченну кількість чистих стратегій має ціну гри.

Тобто, відповідно до теореми Джона фон Неймана отримаємо:

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^T A y = \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^T A y = v.$$

Рівновагою гри в змішаних стратегіях буде пара (x^*, y^*) :

$$\begin{aligned} x^* &= \arg \max_{x \in \Delta_m} \min_{y \in \Delta_n} x^T A y, \\ y^* &= \arg \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^T A y. \end{aligned}$$

Знаходження розв'язку матричної гри в змішаних стратегіях зводиться до розв'язання пари двоїстих задач лінійного програмування

$$\begin{aligned} \max_{x, v} v & & \min_{y, w} w \\ \text{s.t. } A^T x &\geq v \mathbf{1}_n, & \text{s.t. } A y \leq w \mathbf{1}_m, \\ \mathbf{1}_m^T x &= 1, x \geq 0, & \mathbf{1}_n^T y = 1, y \geq 0, \end{aligned}$$

де $\mathbf{1}_l, l \in \{m, n\}$ – вектор з одиниць розмірності l .

Зауважимо, що розв'язок пари двоїстих задач лінійного програмування дає оптимальні змішані стратегії для обох гравців, причому, відповідно до основної теореми двоїстості, оптимальні розв'язки цих задач співпадають:

$$\max_x v = \min_y w.$$

1.3 Біматрична гра

Біматричною грою називають модель зі скінченими множинами стратегій, у якій кожен гравець має власну матрицю виграшів

$$A = (a_{ij})_{m \times n}, B = (b_{ij})_{m \times n},$$

де a_{ij} – виграш першого гравця, а b_{ij} – виграш другого гравця, якщо перший обирає стратегію A_i , а другий – стратегію B_j . На відміну від антагоністичного

випадку, умова $b_{ij} = -a_{ij}$ не вимагається. Сума вигравів може бути як сталою, так і змінною, що дозволяє моделювати конфліктні та частково кооперативні взаємодії.

Наприклад, у ринковій конкуренції дві компанії можуть отримувати різний прибуток залежно від обраних стратегій.

У змішаних стратегіях результат біматричної гри для пари розподілів x та y задається парою значень

$$u_1(x, y) = x^T A y, \quad u_2(x, y) = x^T B y,$$

де u_1 і u_2 – очікувані виграти для першого і другого гравця відповідно.

Метою кожного гравця є максимізація власної функції виграву, враховуючи, що інший гравець діє раціонально.

Означення 1.3

Пара стратегій (x^*, y^*) називається рівновагою за Нешем у змішаних стратегіях за умови

$$\begin{aligned} u_1(x^*, y^*) &\geq u_1(x, y^*) && \forall x, \\ u_2(x^*, y^*) &\geq u_2(x^*, y) && \forall y, \end{aligned}$$

де u_1, u_2 – очікувані виграти.

Тобто якщо жодному з гравців не вигідно відхилитися від своєї рівноважної стратегії, якщо інший гравець притримується своєї рівноважної стратегії.

Концепція рівноваги, незважаючи на її переваги, не завжди описує очікувану поведінку гравців навіть у випадках, коли існує єдина рівновага за Нешем.

На відміну від антагоністичних ігор, в біматричних іграх захисні і урівноважені пари стратегій можуть не співпадати. Розглянемо як визначаються захисні стратегії в біматричних іграх.

Незалежно від дій другого гравця перший гравець може гарантувати собі середній виграш v_1^0 , який дорівнює ціні матричної гри з матрицею A . Аналогічно, незалежно від поведінки першого гравця, другий гравець може гарантувати собі середній виграш v_2^0 , який дорівнює ціні матричної гри з матрицею B^T .

Числа

$$v_1^0 = \max_{x \in \Delta_m} \min_{y \in \Delta_n} u_A(x, y) \text{ і } v_2^0 = \max_{y \in \Delta_n} \min_{x \in \Delta_m} u_B(x, y)$$

називаються *рівнями безпеки* гравців або *гарантованим результатом*.

Стратегія $x^0(y^0)$, що забезпечує гарантований результат першому (другому) гравцю, незалежно від дій супротивника, називається його *обережною (максимінною) стратегією*. Така стратегія особливо важлива у ситуаціях невизначеності або ризику.

РОЗДІЛ 2. Робастна оптимізація

2.1 Поняття про робастну оптимізацію

На практиці навіть незначні збурення в даних можуть зробити розв'язок оптимізаційної задачі не лише не оптимальним, а й недопустимим. Робастна оптимізація пропонує підхід, який дозволяє знаходити рішення, стійкі до таких збурень [5].

Нехай маємо задачу невизначеної лінійної оптимізаційної

$$\left\{ \min_x \left\{ c^T x : Ax \leq d \right\} \right\}_{(c,A,d) \in U} \quad (2.1)$$

де $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$ – невизначені коефіцієнти, а U – задана множина невизначеностей.

Припустимо, що $c \in \mathbb{R}^n$ та $d \in \mathbb{R}^m$ є надійними, тоді робастний аналог (2.1) можемо задати в наступному вигляді:

$$\min_x \{c^T x : A(\xi)x \leq d \quad \forall \xi \in U\} \quad (2.2)$$

де $U \subset \mathbb{R}^L$ — задана множина невизначеностей. Це означає, що розв'язок має бути допустимим для всіх можливих реалізацій параметрів з множини невизначеності.

Можна зосередитися на одному обмеженні, оскільки невизначеність у робастній оптимізації стосується саме обмежень.

Обмеження взяті з (2.2) можемо задати наступним чином

$$(a + P\xi)^T x \leq d \quad \forall \xi \in U. \quad (2.3)$$

Розмірність загального параметра невизначеності $P\xi$ часто набагато вища, ніж у параметра ξ .

Означення 2.1.

Вектор $x \in \mathbb{R}^n$ є робастним допустимий розв'язком для (2.2), якщо від задовільняє всім реалізаціям обмежень з набору невизначеностей, тобто

$$A(\xi)x \leq d \quad \forall \xi \in U.$$

Означення 2.2.

Для заданого кандидатного рішення x , робастне значення $\hat{c}(x)$ для цільової функції (2.1) є найбільшим значенням початкової цільової функції $c^T x$ серед усіх реалізацій даних з набору невизначеностей:

$$\hat{c}(x) = \sup_{(c,A,b) \in u} [c^T x] \quad (2.4)$$

Означення 2.3.

Робастний аналог невизначеної задачі лінійної оптимізації — це задача оптимізації

$$\min_x \left\{ \hat{c}(x) = \sup_{(c,A,b) \in U} [c^T x] : Ax \leq b \quad \forall (c,A,b) \in U \right\} \quad (2.5)$$

надійного значення цільової функції для всіх надійних реалістичних рішень невизначеної задачі.

Розглянемо задачу лінійної оптимізації

$$\begin{aligned} & \max cx \\ & s.t. \sum_j \tilde{a}_{ij} x_j \leq \tilde{b}_i \quad \forall i \end{aligned} \quad (2.6)$$

Розглянемо i -те обмеження, де коефіцієнти та параметри залежать від множини невизначеності. Задаємо невизначеність наступним чином

$$\tilde{a}_{ij} = a_{ij} + \xi_{ij} \hat{a}_{ij} \quad \forall j \in J_i$$

$$\tilde{b}_i = b_i + \xi_{i0} \hat{b}_i$$

де a_{ij}, b_i — номінальні значення параметрів, \hat{a}_{ij}, \hat{b}_i — збурення, J_i — підмножина індексів, коефіцієнти яких залежать від невизначеностей, ξ_{i0} та ξ_{ij} — випадкові величини.

Задачу лінійної оптимізації (2.6) можемо переписати як

$$\sum_{j \notin J_i} a_{ij} x_j + \sum_{j \in J_i} \tilde{a}_{ij} x_j \leq \tilde{b}_i$$

Перепишемо наступним чином

$$\sum_j a_{ij} x_j + \left[\sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j \right] \leq b_i.$$

У методі робастної оптимізації з попередньо визначеним набором невизначеностей U метою є пошук рішень, які є допустимими для будь-якого ξ у заданому наборі невизначеностей. Замінивши вихідне обмеження в задачі лінійної оптимізації (2.6), отримаємо робастний аналог лінійної оптимізації

$$\begin{aligned} & \max cx \\ \text{s.t. } & \sum_j a_{ij} x_j + \max_{\xi \in U} \left\{ \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j \right\} \leq b_i \quad \forall i. \end{aligned} \quad (2.7)$$

РОЗДІЛ 3. Області невизначеності

У звичайній матричній грі елементи a_{ij} матриці виграшів A відомі наперед. Але в реальних ситуаціях матриця A може містити неточності, помилки, випадкові або суб'єктивні дані. Тому замість матриці A будемо розглядати множину матриць $\tilde{A}(\xi), \xi \in U$, де U – множина невизначеності [9]. Робастно допустимий розв'язок повинен задовольняти усім обмеженням задачі для усіх можливих матриць \tilde{A} з множини невизначеності U . Робастна рівновага в матричній грі – це набір стратегій для кожного гравця, які гарантують найкращий результат у найгіршому випадку:

$$x^* = \arg \max_{x \in \Delta_m} \min_{\xi \in U} \min_{y \in \Delta_n} x^T \tilde{A}(\xi) y, \quad y^* = \arg \min_{y \in \Delta_n} \max_{\xi \in U} \max_{x \in \Delta_m} x^T \tilde{A}(\xi) y.$$

Розв'язок робастної задачі суттєво залежить від форми області невизначеності. Цю область часто описують використовуючи метрику Мінковського при різних значення параметра p [8].

$$\|\xi\|_p = \left(\sum_{j \in J_i} |\xi_j|^p \right)^{\frac{1}{p}}, \quad p = 1, 2, \infty.$$

3.1 Прямокутна область невизначеності (Box, Soyster, 1973)

Набір невизначеностей прямокутної області невизначеності [1] описується за допомогою ∞ -норми вектора невизначених даних наступним чином

$$U_\infty = \{ \xi \mid \|\xi\|_\infty \leq \rho \} = \{ \xi \mid |\xi_j| \leq \rho, \forall j \in J_i \} \quad (3.1)$$

де ρ – регульований параметр, що контролює розмір прямокутної множини невизначеності.

Позначимо через J_i – множину індексів i -го рядка, для яких відповідні елементи збуреної матриці \tilde{A} мають невизначеності.

Припустимо, що невизначені параметри обмежені в заданих симетричних інтервалах $\tilde{a}_{ij} \in [a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}] \forall j \in J_i$, тоді невизначеність можемо представити таким чином $\tilde{a}_{ij} = a_{ij} + \xi_j \hat{a}_{ij}$, $\xi_j \in [-1, 1]$. Отримуємо інтервальний набір невизначеностей, що є окремим випадком прямокутної області невизначеності при $\rho = 1$. На рисунку 3.1 зображено множину невизначеності для одного лінійного рівняння з двома змінними і номінальними коефіцієнтами a_1 і a_2 .

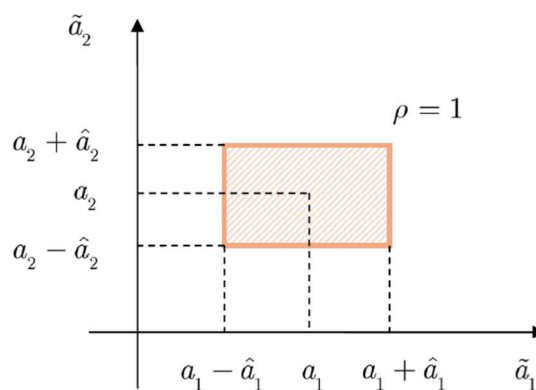


Рис. 3.1. Прямокутна область невизначеності

Розв'язання даного підходу є досить простим, проте він дозволяє всім параметрам одночасно приймати гранично несприятливі значення, тому розв'язки прямокутної області невизначення схильні до зменшення оптимального значення цільової функції.

3.2 Еліпсоїдальна область невизначеності (Ellipsoidal set, Ben-Tal & Nemirovski, 1998)

Еліпсоїдальна область невизначеностей [2] описується за допомогою 2-норми вектора невизначених даних

$$U_2 = \{ \xi \mid \|\xi\|_2 \leq \varepsilon \} = \left\{ \xi \mid \sqrt{\sum_{j \in J_i} \xi_j^2} \leq \varepsilon \right\} \quad (3.2)$$

де ε – регульований параметр, що контролює розмір еліпсоїдальної множини невизначеності.

З геометрії відомо, що для обмеженої невизначеності $\xi_j \in [-1;1]$, коли $\varepsilon \geq \sqrt{|J_i|}$ весь невизначений простір покритий множиною еліпсоїдальної області невизначеностей.

Графічна ілюстрація еліпсоїдальної області для рівняння з двома змінними зображена на рисунку 3.2.

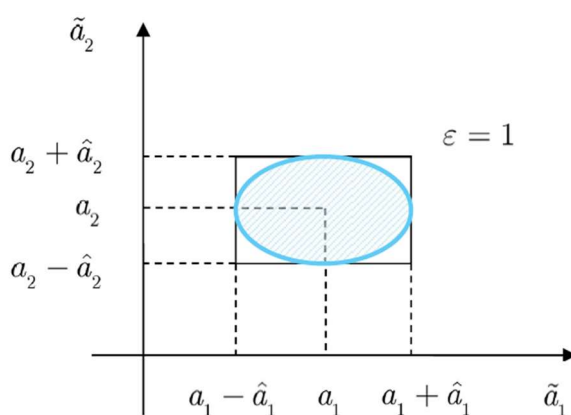


Рис. 3.2. Еліпсоїдальна область невизначеності

Еліпсоїдальна множина дозволяє враховувати кореляції між параметрами та відкидає найгірші результати, тому забезпечує менш консервативне рішення. Робастна задача з еліпсоїдальною областю невизначеності зводиться до задач другого порядку (SOCP).

3.3 Перетин прямокутної та еліпсоїдальної області невизначеності

Перетин прямокутної та еліпсоїдальної областей невизначеності можемо описати наступним чином

$$U_{2 \cap \infty} = \left\{ \xi \mid \sum_{j \in J_i} \xi_j^2 \leq \varepsilon^2, |\xi_j| \leq \rho, \forall_j \in J_i \right\} \quad (3.3)$$

Параметри, які регулюють розмір невизначеності мають задовольняти умову

$$\rho \leq \varepsilon \leq \rho \sqrt{|J_i|}.$$

Зауваження.

При $\varepsilon \leq \rho$ – еліпс вписаний у прямокутник, при $\varepsilon \geq \rho \sqrt{|J_i|}$ – еліпс описаний навколо прямокутника, тобто перетином прямокутника та еліпса є прямокутник.

На рисунку 3.3 зображена графічна ілюстрація перетину прямокутної та еліпсоїдальної області при $\rho = 1, \varepsilon = 1,2$ для лінійного рівняння з двома змінними.

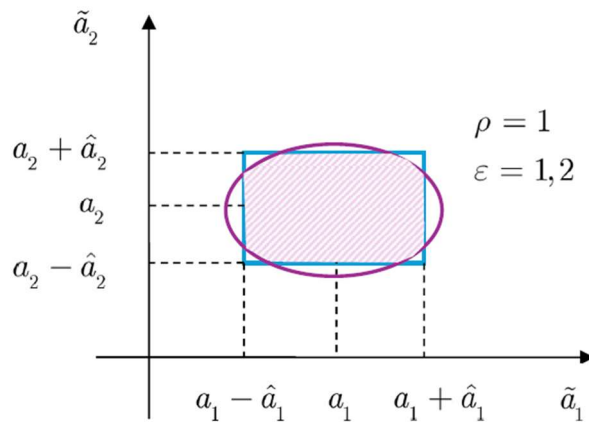


Рис. 3.3. Перетин прямокутної і еліпсоїдальної областей

3.4 Невизначеність в лівих частинах обмежень (LHS)

Якщо в i -му обмеженні (2.6) враховуємо лише невизначеність зліва, тоді (2.7) зводиться до такого вигляду

$$\sum_j a_{ij} x_j + \left[\max_{\xi \in U} \left\{ \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j \right\} \right] \leq b_i \quad (3.4)$$

Властивість 3.1 [5]

Якщо множина U є прямокутною областю невизначеності (3.1), то відповідне робастне обмеження (3.4) матиме наступний вигляд

$$\sum_i a_{ij} x_j + \left[\varepsilon \sum_{j \in J_i} \hat{a}_{ij} |x_j| \right] \leq b_i.$$

Властивість 3.2 [5]

Якщо множина U є еліпсоїдальною областю невизначеності (3.2), то відповідне робастне обмеження (3.4) еквівалентне наступному обмеженню

$$\sum_j a_{ij} x_j + \left[\varepsilon \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 x_j^2} \right] \leq b_i$$

Властивість 3.3 [5]

Якщо множина U є перетином прямокутної та еліпсоїдальної області невизначеності (3.3), то відповідне робастне обмеження (3.4) еквівалентне наступному обмеженню

$$\sum_j a_{ij} x_j + \left[\rho \sum_{j \in J_i} \hat{a}_{ij} |x_j - z_{ij}| + \varepsilon \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2} \right] \leq b_i. \quad (3.5)$$

Доведення.

Множину невизначеності перетину прямокутної та еліпсоїдальної області (3.3) позначимо за допомогою кінчного представлення наступним чином

$$U_{2 \cap \infty} = \left\{ \xi \mid P_2 \xi + p_2 \in K_2, P_\infty \xi + p_\infty \in K_\infty \right\},$$

де $K_2 = \left\{ [\theta_{L \times 1}; t] \in R^{L+1} \mid \|\theta\|_2 \leq t \right\}$, $K_\infty = \left\{ [\theta_{L \times 1}; t] \in R^{L+1} \mid \|\theta\|_\infty \leq t \right\}$. Задача

внутрішньої оптимізації матиме наступний вигляд

$$\max_{\xi} \left\{ \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j : P_2 \xi + p_2 \in K_2, P_\infty \xi + p_\infty \in K_\infty \right\}.$$

Визначення подвійної змінної $y^1 = [w_i, \tau_1] \in R^{L+1}, y^2 = [z_i, \tau_2] \in R^{L+1}$ та використовуючи подвійний конус $K_\infty^* = K_1, K_2^* = K_2$ конічний двоїсту задачу внутрішньої максимізації виглядатиме так:

$$\min \left\{ \rho \tau_1 + \varepsilon \tau_2 : w_i + z_i = \hat{a}_i x, \|w_i\|_1 \leq \tau_1, \|z_i\|_2 \leq \tau_2 \right\}.$$

Після еквівалентного перетворення шляхом заміни τ_1 та τ_2 на $\|w_i\|_1 = \sum_{j \in J_i} |w_{ij}|, \|z_i\|_2 = \sqrt{\sum_{j \in J_i} z_{ij}^2}$, отримаємо

$$\min_{z, w} \left\{ \rho \sum_{j \in J_i} |w_{ij}| + \varepsilon \sqrt{\sum_{j \in J_i} z_{ij}^2} : w_i + z_i = \hat{a}_i x \right\},$$

що еквівалентне

$$\min_z \rho \sum_{j \in J_i} |\hat{a}_{ij} x_j - z_{ij}| + \varepsilon \sqrt{\sum_{j \in J_i} z_{ij}^2}.$$

Оскільки z_{ij} змінні прийняття рішень, замінемо на $\hat{a}_{ij} z_{ij}$, отже еквівалентна задача має наступний вигляд

$$\min_z \rho \sum_{j \in J_i} |\hat{a}_{ij} x_j - \hat{a}_{ij} z_{ij}| + \varepsilon \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2}.$$

Включивши вище зазначену конічну двоїсту функцію в робастний аналог обмеження та видаливши оператор мінімізації, отримаємо наступний робастний аналог

$$\sum_j a_{ij} x_j + \left[\rho \sum_{j \in J_i} \hat{a}_{ij} |x_j - z_{ij}| + \varepsilon \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2} \right] \leq b_i.$$

Зауваження.

Еквівалентне робастне обмеження для (3.5) можемо отримати замінивши член абсолютного значення $|x_j - z_{ij}|$ допоміжною змінною u_{ij} наступним чином

$$\begin{cases} \sum_j a_{ij} x_j + \rho \sum_{j \in J_i} \hat{a}_{ij} u_{ij} + \varepsilon \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2} \leq b_i, \\ -u_{ij} \leq x_j - z_{ij} \leq u_{ij}. \end{cases}$$

РОЗДІЛ 4. Перевірка методу

Розглянемо матричну гру, задану матрицею A розміром 2×3 , елементи якої визначають виграш першого гравця для кожного профілю стратегій

$$A = \begin{pmatrix} 9 & 3 & 1 \\ 1 & 2 & 5 \end{pmatrix}.$$

Для розв'язання даної матричної гри як пари двоїстих задач лінійного програмування будемо використовувати мову програмування Python та бібліотеки `numpy`, `scipy`.

1. Номінальна задача.

Для номінальної задачі пара двоїстих задач буде мати наступний вигляд:

$$\begin{array}{ll} \max v & \min w \\ 9x_1 + x_2 \geq v & 9y_1 + 3y_2 + y_3 \leq w \\ 3x_1 + 2x_2 \geq v & y_1 + 2y_2 + 5y_3 \leq w \\ x_1 + 5x_2 \geq v & y_1 + y_2 + y_3 = 1 \\ x_1 + x_2 = 1, & y_1 \geq 0, y_2 \geq 0, y_3 \geq 0. \\ x_1 \geq 0, x_2 \geq 0. & \end{array}$$

Розв'язавши, ми отримали такі результати:

$$\begin{aligned} x_1 &= 0.600, x_2 = 0.400, v = 2.600, \\ y_1 &= 0.000, y_2 = 0.800, y_3 = 0.200, w = 2.600. \end{aligned}$$

Отримані результати x_1, x_2, y_1, y_2, y_3 визначають оптимальні змішані стратегії гравців для заданої матриці виграшів, а значення v, w відповідають максимальному гарантованому виграшу гравців.

На рисунку 4.1 зображено графічний розв'язок пари двоїстих задач лінійного програмування для першого гравця, адже матриця 2×3 дозволяє знайти розв'язок графічно.

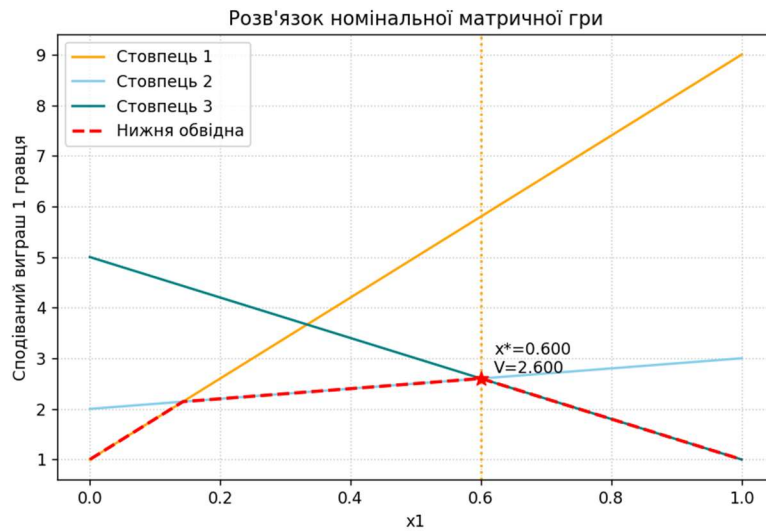


Рис. 4.1. Розв'язок матричної гри з детермінованою матрицею

2. Прямокутна область невизначеності.

Для прямокутної області невизначеності пара двоїстих задач матиме такий вигляд:

$$\begin{array}{ll}
 \max v & \min w \\
 9x_1 + x_2 - \rho(0, 9x_1 + 0, 1x_2) \geq v & 9y_1 + 3y_2 + y_3 + \rho(0, 9y_1 + 0, 3y_2 + 0, 1y_3) \leq w \\
 3x_1 + 2x_2 - \rho(0, 3x_1 + 0, 2x_2) \geq v & y_1 + 2y_2 + 5y_3 + \rho(0, 1y_1 + 0, 2y_2 + 0, 5y_3) \leq w \\
 x_1 + 5x_2 - \rho(0, 1x_1 + 0, 5x_2) \geq v & y_1 + y_2 + y_3 = 1 \\
 x_1 + x_2 = 1, & y_1 \geq 0, y_2 \geq 0, y_3 \geq 0. \\
 x_1 \geq 0, x_2 \geq 0. &
 \end{array}$$

Розв'язавши, отримаємо результати:

$$x_1 = 0.600, x_2 = 0.400, v = 2.340,$$

$$y_1 = 0.000, y_2 = 0.800, y_3 = 0.200, w = 2,860.$$

Отримані стратегії першого та другого гравця демонструють, що для прямокутної області невизначеності гарантується виграш $v = 2.340$ та $w = 2.860$.

На рисунку 4.2 зображено графічний розв'язок для першого гравця.

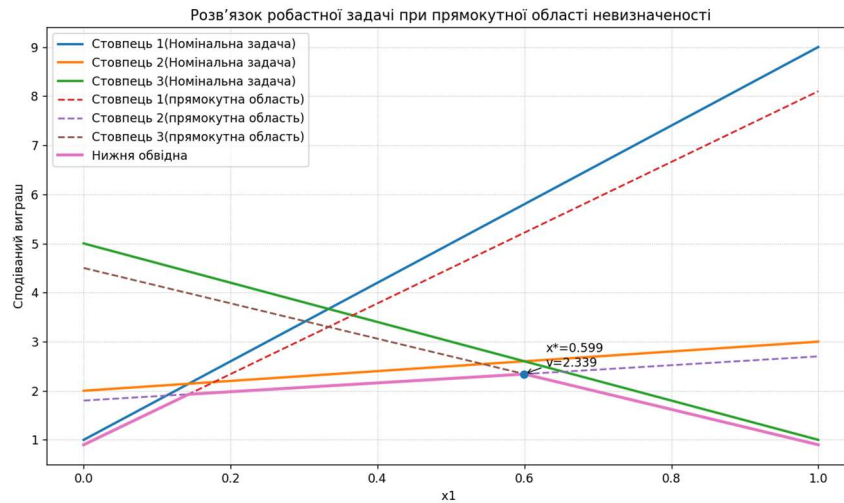


Рис. 4.2. Розв'язок матричної гри для прямокутної області невизначеності

3. Еліпсоїдальна область невизначеності.

Для такого типу невизначеності пара двоїстих задач виглядатиме так:

$$\begin{aligned}
 & \max v \\
 & 9x_1 + x_2 - \varepsilon \sqrt{(0,9x_1)^2 + (0,1x_2)^2} \geq v \\
 & 3x_1 + 2x_2 - \varepsilon \sqrt{(0,3x_1)^2 + (0,2x_2)^2} \geq v \\
 & x_1 + 5x_2 - \varepsilon \sqrt{(0,1x_1)^2 + (0,5x_2)^2} \geq v \\
 & x_1 + x_2 = 1, \\
 & x_1 \geq 0, x_2 \geq 0.
 \end{aligned}$$

$$\begin{aligned}
 & \min w \\
 & 9y_1 + 3y_2 + y_3 + \varepsilon \sqrt{(0,9y_1)^2 + (0,3y_2)^2 + (0,1y_3)^2} \leq w \\
 & y_1 + 2y_2 + 5y_3 + \varepsilon \sqrt{(0,1y_1)^2 + (0,2y_2)^2 + (0,5y_3)^2} \leq w \\
 & y_1 + y_2 + y_3 = 1 \\
 & y_1 \geq 0, y_2 \geq 0, y_3 \geq 0.
 \end{aligned}$$

Отримали наступні результати:

$$x_1 = 0.597, x_2 = 0.403, v = 2.401,$$

$$y_1 = 0.000, y_2 = 0.790, y_3 = 0.210, w = 2.819.$$

Отримані стратегії першого та другого гравця демонструють, що при еліпсоїдальній області невизначеності гарантується вигреш $v = 2.401$ та $w = 2.819$.

На рисунку 4.3 зображено графічний розв'язок для першого гравця.

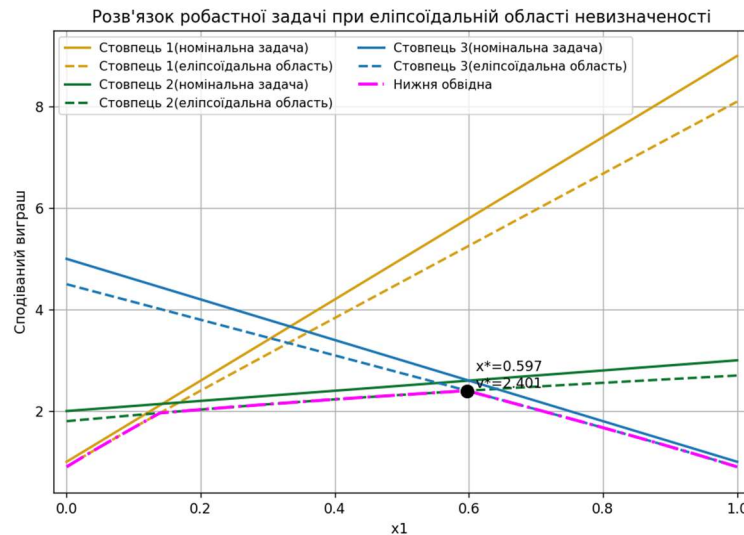


Рис. 4.3. Розв'язок матричної гри для еліпсоїдальної області невизначеності

4. Перетин прямокутної та еліпсоїдальної області невизначеності.

Для даного типу невизначеності пара двоїстих задач виглядатиме таким

чином:

$$\begin{aligned} & \max v \\ & 9x_1 + x_2 - \rho(0,9u_{11} + 0,1u_{21}) - \varepsilon\sqrt{(0,9z_{11})^2 + (0,1z_{21})^2} \geq v \\ & 3x_1 + 2x_2 - \rho(0,3u_{12} + 0,2u_{22}) - \varepsilon\sqrt{(0,3z_{12})^2 + (0,2z_{22})^2} \geq v \\ & x_1 + 5x_2 - \rho(0,1u_{13} + 0,5u_{23}) - \varepsilon\sqrt{(0,1z_{13})^2 + (0,5z_{23})^2} \geq v \\ & -u_{11} \leq x_1 - z_{11} \leq u_{11}, -u_{21} \leq x_2 - z_{21} \leq u_{21} \\ & -u_{12} \leq x_1 - z_{12} \leq u_{12}, -u_{22} \leq x_2 - z_{22} \leq u_{22} \\ & -u_{13} \leq x_1 - z_{13} \leq u_{13}, -u_{23} \leq x_2 - z_{23} \leq u_{23} \\ & x_1 + x_2 = 1, \\ & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$

$$\begin{aligned} & \min w \\ & 9y_1 + 3y_2 + y_3 + \rho(0,9u_{11} + 0,3u_{12} + 0,1u_{13}) + \varepsilon\sqrt{(0,9z_{11})^2 + (0,3z_{12})^2 + (0,1z_{13})^2} \leq w \\ & y_1 + 2y_2 + 5y_3 + \rho(0,1u_{21} + 0,2u_{22} + 0,5u_{23}) + \varepsilon\sqrt{(0,1z_{21})^2 + (0,2z_{22})^2 + (0,5z_{23})^2} \leq w \\ & -u_{11} \leq y_1 - z_{11} \leq u_{11}, -u_{12} \leq y_2 - z_{12} \leq u_{12}, -u_{13} \leq y_3 - z_{13} \leq u_{13}, \\ & -u_{21} \leq y_1 - z_{21} \leq u_{21}, -u_{22} \leq y_2 - z_{22} \leq u_{22}, -u_{23} \leq y_3 - z_{23} \leq u_{23}, \\ & y_1 + y_2 + y_3 = 1 \\ & y_1 \geq 0, y_2 \geq 0, y_3 \geq 0. \end{aligned}$$

Отримані результати:

$$\begin{aligned} x_1 = 0.624, x_2 = 0.376, v = 2.385, \\ y_1 = 0.000, y_2 = 0.800, y_3 = 0.200, w = 2.859. \end{aligned}$$

Отримані стратегії першого та другого гравця демонструють, що при еліпсоїдальній області невизначеності гарантується вигравш $v = 2.385$ та $w = 2.859$.

Побудувати графічний розв'язок вже не є можливим, адже дана задача має більше ніж дві змінні в системі обмежень.

5. Порівняння результатів.

Для зручності результати виведемо в таблицю 4.1.

Таблиця 4.1. Результати розв'язків матричної гри

	Номинальна задача	Прямокутна область невизначеності	Еліпсоїдальна область невизначеності	Перетин прямокутної та еліпсоїдальної області
x_1	0,6	0,6	0,6	0,62
x_2	0,4	0,4	0,4	0,38
v	2,6	2,34	2,4	2,39
y_1	0	0	0	0
y_2	0,8	0,8	0,8	0,8
y_3	0,2	0,2	0,2	0,2
w	2,6	2,86	2,8	2,86

У даному розділі було перевірено як працює метод робастної оптимізації на простішій тестовій моделі. Така перевірка необхідна була для того, щоб впевнитись, що чисельні алгоритми реалізуються коректно.

Отримані результати номінальної задачі підтверджують наявність розв'язку у змішаних стратегіях, а також побудовані захисні стратегії забезпечують гравцям гарантований виграш незалежно від вибору стратегії конкурента. Після цього було розглянуто три типи невизначеності, відповідно отримані результати показали, що оптимальні стратегії гравців та значення гарантованого виграшу залежить від типу та розмірності області невизначеностей. Зазначимо, що розв'язки двоїстих задач зі збуреними даними, на відміну від детермінованої матричної гри, вже не співпадають, оскільки найгірші сценарії для кожного з гравців можуть бути різними.

РОЗДІЛ 5. Оптимізація ігрової моделі ринкової конкуренції

5.1 Постановка завдання дослідження

Складемо ігрову математичну модель ринкової конкуренції. Відповідно до інформації з відкритих джерел [10], дві великих торгових мережі в 2024 році мали майже однаковий виторг: Novus (ТОВ «Новус Україна») – 29 млрд грн, а Metro Cash&Carry (ТОВ «Метро кеш енд кері Україна») – 28,7 млрд грн. Маючи диференційовану структуру попиту, ці ритейли не можна вважати антагоністами, тому математичну модель цінової конкуренції складемо у вигляді біматричної гри.

Кожен гравець має 10 стратегій – діапазон цін відносно базової ціни p . Платежем в кожній матриці гри для кожного профілю стратегій вважаємо прибуток в млн грн, який можуть отримати гравці.

Вхідні дані:

1. Новус (гравець А) – виторг 29 млрд грн, Метро (гравець В) – 28,7 млрд грн.
2. Собівартість – 85% (витрати на закупівлю, заробітну плату тощо).
3. Маржа – 15% (різниця між ціною продажу і собівартістю).
4. Діапазон цін – 0.90, 0.92, 0.94, 0.96, 0.98, 1.00, 1.02, 1.04, 1.06 і 1.08.
5. Вважатимемо попит сегментованим відносно чутливості покупців до змінення ціни: локальні покупці, які віддають переваги найближчому магазину; оптові, які здійснюють покупки регулярно і великими обсягами; онлайн, які здійснюють покупки через інтернет, преміум, які цінують якість і готові платити більше; покупці, чутливі до ціни (діскаунти) (табл. 5.1).

Таблиця 5.1. Сегментація попиту

№	Сегмент	Частка Новус	Частка Метро	Лояльність Новус	Лояльність Метро	Еластичність Новус	Еластичність Метро
1	Локальні	40%	20%	0.65	0.45	1.30	1.60
2	Оптовики	15%	45%	0.60	0.80	1.00	0.75
3	Інтернет	20%	15%	0.35	0.25	1.90	2.20
4	Преміум	15%	10%	0.70	0.55	0.55	0.70
5	Дискаунт	10%	10%	0.25	0.25	2.60	2.40

Лояльність – частка клієнтів, які залишаються при змінній ціні, еластичність попиту – чутливість попиту до змінення ціни.

Прибуток для кожного гравця будемо знаходити за формулою:

$$\pi = \max \begin{cases} \sum_{i=1}^5 \pi_i + TE + \omega, \\ -150, \end{cases}$$

де $\pi_i, i = \overline{1,5}$ – прибуток кожного сегменту, TE – загальний ефект, який не залежить від сегменту і враховує взаємодію обох гравців на рівні всього ринку: штрафи за цінову війну (демпінг), бонус за цінову диференціацію, штраф за несправедливу конкуренцію, ω – ринкові коливання в діапазоні ± 55 .

Наведемо розрахунок виплат для першого гравця. Для конкурента виплати обчислюються аналогічно.

1) Частка сегменту s_A в залежності від власної ціни p_A і ціни конкурента p_B :

$$s_A = s_0 - 0,6(1 - l_A)(p_A - p_B),$$

де s_0 – базова частка, l_A – частка сегменту, який залишається (лояльність покупців), коефіцієнт 0,6 регулює швидкість переходу клієнтів.

2) Запровадимо асиметричний фактор попиту для різних рівнів цін в залежності від еластичності попиту ε_A :

$$d_A = \begin{cases} 1 - 12\varepsilon_A (p_A - 1,02)^2, & p_A \geq 1,02; \\ 1 + 0,8\varepsilon_A |p_A - 0,98|, & p_A \leq 0,98; \\ 1,05, & 0,98 < p_A < 1,02. \end{cases}$$

3) Штраф за демпінгування цін:

$$f_A^1 = 300R_A^0 (0,96 - p_A)^2, \text{ якщо } p_A < 0,96$$

$$f_A^2 = 25R_A^0 \left[(0,99 - p_A)(0,99 - p_B) \right]^{\frac{3}{2}}, \text{ якщо } p_A < 0,99, p_B < 0,99,$$

$$f_A = f_A^1 + f_A^2,$$

де R_A^0 – базовий виторг першого гравця (вхідна інформація з відкритих джерел).

4) Врахуємо ефект конкуренції c , який впливає на прибуток. При близьких цінах у гравців прибуток знижується (жорстка конкуренція), при великій різниці – також знижується за рахунок втрати клієнтів.

Позначимо $\Delta p = |p_A - p_B|$. Нехай

$$\Delta p < 0,015 \quad \Rightarrow c_A = c_B = 0,65;$$

$$0,015 \leq \Delta p < 0,035 \quad \Rightarrow c_A = c_B = 0,75;$$

$$0,035 \leq \Delta p < 0,07 \quad \Rightarrow c_A = c_B = 1;$$

$$0,07 \leq \Delta p < 0,12 \quad \Rightarrow c_A = 0,9, c_B = 0,7 (p_A < p_B);$$

$$\Delta p \geq 0,12 \quad \Rightarrow c_A = 0,85, c_B = 0,6 (p_A < p_B).$$

5) Нехай оптимальна ціна для кожного сегменту попиту задана в таблиці 4.2.

Таблиця 5.2. Оптимальні ціни для кожного сегменту

№	Сегмент	Оптимальна ціна p_{opt} (відносно базової) Новус	Оптимальна ціна p_{opt} (відносно базової) Метро
1	Локальні	1,03	0,99
2	Оптовики	0,97	1,01
3	Інтернет	1,05	1,02
4	Преміум	1,06	1,04
5	Дискаунт	0,93	0,95

Визначимо коефіцієнт оптимальності встановленої першим гравцем ціни для кожного сегменту за формулою:

$$k_A^{opt} = e^{-\left(\frac{p_A - p_{opt}}{0,04}\right)^2}.$$

7) Реальна маржа:

$$m_A = 0,15(p - 0,85);$$

обсяг продажів сегменту з усіма коригуваннями:

$$R_A = R_A^0 s_A d_A k_A^{opt} c_A;$$

прибуток сегменту:

$$\pi_{Ai} = 1000R_A m_A - f_A \text{ (млн. грн).}$$

Задамо штраф за «цінову війну», якщо обидва гравці встановлюють ціни $p < 0,95$:

$$TE = -60R_A^0 \left[(0,95 - p_A)(0,95 - p_B) \right]^2.$$

Округливши отримані значення, отримаємо платіжні матриці – матриці прибутків для обох гравців.

Виплати гравця А

Метро Новус	0,9	0,92	0,94	0,96	0,98	1,00	1,02	1,04	1,06	1,08
0,90	-150	-150	-150	-150	-150	-150	-150	-150	-150	-150
0,92	-54	-48	-76	17	-27	5	-59	-14	-46	19
0,94	101	29	23	21	141	60	99	19	108	60
0,96	128	121	67	62	83	190	98	127	82	159
0,98	101	166	140	105	96	132	200	140	140	157
1,00	249	200	251	250	208	172	226	275	279	233
1,02	339	366	348	392	434	350	272	352	431	487
1,04	398	468	432	466	511	589	450	347	449	589
1,06	372	339	418	353	421	462	541	381	307	394
1,08	148	244	186	238	190	262	278	337	198	192

Виплати гравця В

Метро Новус	0,9	0,92	0,94	0,96	0,98	1,00	1,02	1,04	1,06	1,08
0,90	-150	-67	92	171	237	458	419	268	159	2
0,92	-150	-60	22	166	322	419	454	335	113	87
0,94	-150	-91	17	103	301	502	444	297	181	19
0,96	-150	-4	13	93	233	509	506	330	108	62
0,98	-150	-50	130	122	208	413	557	364	172	10
1,00	-150	-16	49	243	268	351	450	443	180	75
1,02	-150	-81	89	152	381	443	361	338	255	65
1,04	-150	-30	14	182	327	566	462	253	157	119
1,06	-150	-62	104	139	312	578	579	338	115	27
1,08	-150	4	57	217	331	508	641	445	163	44

Оскільки поведінка конкурента невідома, будемо знаходити захисні стратегії кожного гравця.

Нехай $x = (x_1, \dots, x_{10})^T$ – захисна стратегія гравця A , $y = (y_1, \dots, y_{10})^T$ – захисна стратегія гравця B ,

$$x \in \Delta_{10} = \left\{ x \geq 0, \sum_{i=1}^{10} x_i = 1 \right\}, y \in \Delta_{10} = \left\{ y \geq 0, \sum_{j=1}^{10} y_j = 1 \right\}.$$

Тоді

$$x^* = \arg \max_{x \in \Delta_{10}} \min_{y \in \Delta_{10}} x^T A y, \quad y^* = \arg \max_{y \in \Delta_{10}} \min_{x \in \Delta_{10}} x^T B^T y.$$

Елементи матриць виплат достеменно невідомі, тому будемо знаходити розв'язок робастного аналога моделі

$$x^* = \arg \max_{x \in \Delta_{10}} \min_{\xi \in U} \min_{y \in \Delta_{10}} x^T \tilde{A}(\xi) y, \quad y^* = \arg \max_{y \in \Delta_{10}} \min_{\xi \in U} \min_{x \in \Delta_{10}} x^T \tilde{B}^T(\xi) y.$$

Гарантований результат і захисні стратегії гравців знаходимо, розв'язуючи пару задач:

$$\begin{aligned} \max_{x,v} v & \quad \max_{y,w} w \\ \sum_i a_{ij} x_i + \min_{\xi \in U} \left\{ \sum_i \xi_{ij} \hat{a}_{ij} x_i \right\} \geq v, \forall j & \quad \sum_j b_{ij} y_j + \min_{\xi \in U} \left\{ \sum_j \xi_{ij} \hat{b}_{ij} y_j \right\} \geq w, \forall i \\ \sum_i x_i = 1, x_i \geq 0, \forall i & \quad \sum_j y_j = 1, y_j \geq 0, \forall j \end{aligned}$$

Для різних типів геометрії областей невизначеності розв'язок знаходимо розв'язуючи наступні задачі.

Прямокутна область:

$$\begin{aligned} \max_{x,v} v & \quad \max_{y,w} w \\ \sum_i a_{ij} x_i - \rho \sum_i |\hat{a}_{ij}| x_i \geq v, \forall j & \quad \sum_j b_{ij} y_j - \rho \sum_j |\hat{b}_{ij}| y_j \geq w, \forall i \\ \sum_i x_i = 1, x_i \geq 0, \forall i & \quad \sum_j y_j = 1, y_j \geq 0, \forall j. \end{aligned}$$

Еліпсоїдальна область:

$$\begin{aligned} \sum_i a_{ij} x_i - \varepsilon \sqrt{\sum_i \hat{a}_{ij}^2 x_i^2} &\geq v, \forall j & \sum_j b_{ij} y_j - \varepsilon \sqrt{\sum_j \hat{b}_{ij}^2 y_j^2} &\geq w, \forall i \\ \sum_i x_i &= 1, x_i \geq 0, \forall i & \sum_j y_j &= 1, y_j \geq 0, \forall j. \end{aligned}$$

Перетин прямокутної і еліпсоїдальної областей:

$$\begin{aligned} \sum_i a_{ij} x_i - \rho \sum_i \hat{a}_{ij} u_{ij} - \varepsilon \sqrt{\sum_i \hat{a}_{ij}^2 z_{ij}^2} &\geq v, \forall j \\ -u_{ij} &\leq x_i - z_{ij} \leq u_{ij} \\ \sum_i x_i &= 1, x_i \geq 0, \forall i, \end{aligned}$$

$$\begin{aligned} \sum_j b_{ij} y_j - \rho \sum_j \hat{b}_{ij} u_{ij} - \varepsilon \sqrt{\sum_j \hat{b}_{ij}^2 z_{ij}^2} &\geq w, \forall i \\ -u_{ij} &\leq y_j - z_{ij} \leq u_{ij} \\ \sum_j y_j &= 1, y_j \geq 0, \forall j. \end{aligned}$$

5.2 Результати обчислень

Для кожної матриці перевіримо існування розв'язку в чистих стратегіях.

Для матриці A :

$$\max_i \min_j = 347, \min_j \max_i = 381.$$

Для матриці B^T :

$$\max_i \min_j = 361, \min_j \max_i = 443.$$

Оскільки для обох гравців $\max_i \min_j \neq \min_j \max_i$, то розв'язку гри в чистих стратегіях немає, тому будемо шукати розв'язок у змішаному розширенні гри.

Для розв'язання біматричної гри в змішаних стратегіях використаємо мову програмування Python. Чисельна реалізація базується на використанні бібліотек

numpy, scipy та cvxpy. Бібліотеку numpy використовують для роботи з масивами, векторними та матричними операціями. Саме завдяки їй виконується більша частина обчислень у матрицях виграшів та побудові змішаних стратегій. Scipy використовується для розв'язання задач оптимізації. Для розв'язання задачі виду SOCP, які виникають у моделюванні еліпсоїдальної області невизначеності, використовували бібліотеку cvxpy. Отримані результати винесені в таблицю 5.3.

Таблиця 5.3. Результати обчислень

	Номінальна задача	Прямокутна область невизначеності	Еліпсоїдальна область невизначеності	Перетин прямокутної та еліпсоїдальної областей невизначеності
Параметр	$\rho = 0; \varepsilon = 0$	$\rho = 1$	$\varepsilon = 1$	$\rho = 1; \varepsilon = 1, 2$
v	366,865	330,179	343,268	338,489
w	401,32	361,188	373,564	368,379
x_1	0	0	0	0
x_2	0	0	0	0
x_3	0	0	0	0
x_4	0	0	0	0
x_5	0	0	0	0
x_6	0	0	0	0
x_7	0,289	0,289	0,253	0,252
x_8	0,169	0,169	0,189	0,188
x_9	0,542	0,542	0,558	0,56
x_{10}	0	0	0	0
y_1	0	0	0	0
y_2	0	0	0	0
y_3	0	0	0	0
y_4	0	0	0	0
y_5	0	0	0	0
y_6	0,492	0,492	0,51	0,515
y_7	0,508	0,508	0,375	0,353
y_8	0	0	0,115	0,132
y_9	0	0	0	0
y_{10}	0	0	0	0

Зауважимо, що для перетину прямокутної та еліпсоїдальної області оптимізації параметри обмежені наступним чином

$$\rho \leq \varepsilon \leq \rho \sqrt{|J_i|},$$

тобто $1 \leq \varepsilon \leq 3,16$. Тому для перетину прямокутної та еліпсоїдальної областей розглянемо розв'язок при різних ε (табл. 5.4).

Таблиця 5.4. Результати обчислень для різних значень параметра ε

	$\rho = 1;$ $\varepsilon = 1,3$	$\rho = 1;$ $\varepsilon = 1,6$	$\rho = 1;$ $\varepsilon = 1,9$	$\rho = 1;$ $\varepsilon = 2,2$	$\rho = 1;$ $\varepsilon = 2,5$	$\rho = 1;$ $\varepsilon = 2,8$	$\rho = 1;$ $\varepsilon = 3,1$
v	336,091	329,124	322,542	316,162	309,894	303,693	297,533
w	365,795	358,051	350,284	342,93	335,906	328,88	321,837
x_1	0	0	0	0	0	0	0
x_2	0	0	0	0	0	0	0
x_3	0	0	0	0	0	0	0
x_4	0	0	0	0	0	0	0
x_5	0	0	0	0	0	0	0
x_6	0	0	0	0	0	0	0
x_7	0,251	0,255	0,264	0,269	0,274	0,279	0,281
x_8	0,187	0,226	0,255	0,274	0,287	0,297	0,305
x_9	0,562	0,499	0,453	0,424	0,408	0,387	0,376
x_{10}	0	0,02	0,028	0,033	0,036	0,037	0,038
y_1	0	0	0	0	0	0	0
y_2	0	0	0	0	0	0	0
y_3	0	0	0	0	0	0	0
y_4	0	0	0	0	0	0	0
y_5	0	0	0,02	0,062	0,069	0,074	0,077
y_6	0,517	0,523	0,524	0,455	0,446	0,44	0,435
y_7	0,345	0,326	0,306	0,299	0,312	0,323	0,333
y_8	0,138	0,151	0,15	0,184	0,173	0,163	0,155
y_9	0	0	0	0	0	0	0
y_{10}	0	0	0	0	0	0	0

При фіксованому $\rho = 1$ і зростанні параметра ε у дозволеному інтервалі спостерігається монотонне зменшення гарантованих результатів для обох гравців. Це зумовлене тим, що при збільшенні ε множина невизначеності розширюється, включаючи несприятливі сценарії.

Виконаємо порівняння результатів (табл. 5.5).

Таблиця 5.5. Результати обчислень

	Номинальна задача	Прямокутна область невизначеності	%	Еліпсоїдальна область невизначеності	%	Перетин прямокутної та еліпсоїдальної областей невизначеності	%
<i>v</i>	366,865	330,179	9,99	343,268	6,43	338,489	7,73
<i>w</i>	401,32	361,188	10	373,564	6,92	368,379	8,2

Для розглянутих параметрів найменше відхилення від розв'язку детермінованої задачі дає еліпсоїдальна область невизначеності.

Висновки

1. Побудовано математичну модель ринкової конкуренції у вигляді біматричної гри, для якої визначено захисні стратегії та гарантований прибуток для кожного гравця у номінальній задачі.

2. Припускаючи, що вхідні дані задачі можуть містити неточності, розроблено робастні аналоги моделі для номінальних матриць моделі для різних множин невизначеностей (прямокутна, еліпсоїдальна, їхній перетин).

3. Знайдено розв'язки робастних оптимізаційних задач для кожного типу області невизначеності, що дозволило порівняти вплив різних сценаріїв на оптимальні стратегії гравців.

4. Чисельна реалізація розв'язку проведена з використанням бібліотек `numpy`, `scipy` та `cvxpy` мови програмування Python, що забезпечило ефективність обчислень.

5. Запропонована методика апробована на модельному прикладі, який дозволяє графічну інтерпретацію результатів.

6. Аналіз отриманих результатів підтверджує, що застосування робастної оптимізації у біматричних ігрових моделях ринкової конкуренції забезпечує баланс між прибутковістю та ризиком, а також є ефективним інструментом для формування цінових стратегій в умовах невизначеності.

Джерела

1. Soyster A. L. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*. 1973. Vol. 21, no. 5. P. 1154–1157. <https://doi.org/10.1287/opre.21.5.1154>
2. Ben-Tal, A., El Ghaoui A.L. and Nemirovski A.: Robust optimization. *Princeton University Press*. 2009. 576 p. DOI:[10.1515/9781400831050](https://doi.org/10.1515/9781400831050)
3. Aghassi M., Bertsimas D. Robust game theory. *Mathematical Programming*. 2006. Vol. 107, no. 1-2. P. 231–273. <https://doi.org/10.1007/s10107-005-0686-0>
4. Crespi G. P., Radi D., Rocca M. Insights on the Theory of Robust Games. *Computational Economics*. 2025. Vol. 65. P. 717–761. <https://doi.org/10.1007/s10614-023-10486-0>
5. Li Z., Ding R., Floudas C. A. A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: I. Robust Linear Optimization and Robust Mixed Integer Linear Optimization. *Industrial & Engineering Chemistry Research*. 2011. Vol. 50, no. 18. P. 10567 –10603. <https://doi.org/10.1021/ie200150p>
6. Gorissen B. L., Yanıkoğlu İ., den Hertog D. A practical guide to robust optimization. *Omega*. 2015. Vol. 53. P. 124–137. <https://doi.org/10.1016/j.omega.2014.12.006>
7. Maschler M., Solan E., Zamir S. Game theory. *Cambridge University Press*. 2013. 979 p.
8. Алексеєва І.В., Перевознюк Т.І. Застосування робастної оптимізації для лінійної моделі функціонування малого підприємства. *Mathematics in Modern Technical University*. 2018, No. 1, P. 61–73.
9. Алексеєва І. В., Сатир Я. М. Робастний підхід до розв’язання матричних ігор з невизначеностями . *XX Міжнародна наукова конференція імені академіка Михайла Кравчука: тези доп., 17–20 листопада 2025 р.* КПІ ім. Ігоря Сікорського. Київ, 2025. С. 45–46.
10. <https://rau.ua/news/analitika-forbes-dohodi-merezh/>

Додаток 1. Гарантований результат та захисні стратегії для біматричної гри змішаних стратегій.

```
import numpy as np
from scipy.optimize import linprog
A = np.array([
    [-150,-150,-150,-150,-150,-150,-150,-150,-150],
    [-54,-48,-76,17,-27,5,-59,-14,-46,19],
    [101,29,23,21,141,60,99,19,108,60],
    [128,121,67,62,83,190,98,127,82,159],
    [101,166,140,105,96,132,200,140,140,157],
    [249,200,251,250,208,172,226,275,279,233],
    [339,366,348,392,434,350,272,352,431,487],
    [398,468,432,466,511,589,450,347,449,589],
    [372,339,418,353,421,462,541,381,307,394],
    [148,244,186,238,190,262,278,337,198,192]
])
B = np.array([
    [-150,-67, 92,171,237,458,419,268,159, 2],
    [-150,-60, 22,166,322,419,454,335,113, 87],
    [-150,-91, 17,103,301,502,444,297,181, 19],
    [-150, -4, 13, 93,233,509,506,330,108, 62],
    [-150,-50,130,122,208,413,557,364,172, 10],
    [-150,-16, 49,243,268,351,450,443,180, 75],
    [-150,-81, 89,152,381,443,361,338,255, 65],
    [-150,-30, 14,182,327,566,462,253,157,119],
    [-150,-62,104,139,312,578,579,338,115, 27],
    [-150, 4, 57,217,331,508,641,445,163, 44]
])
```

```

def defensive_row_player(P):
    m, n = P.shape
    c = np.zeros(m + 1)
    c[-1] = -1.0
    A_ub = np.zeros((n, m + 1))
    b_ub = np.zeros(n)
    for j in range(n):
        A_ub[j, :m] = -P[:, j]
        A_ub[j, -1] = 1.0
    A_eq = np.zeros((1, m + 1))
    A_eq[0, :m] = 1.0
    b_eq = np.array([1.0])
    bounds = [(0.0, 1.0)] * m + [(None, None)]
    res = linprog(c, A_ub=A_ub, b_ub=b_ub,
                 A_eq=A_eq, b_eq=b_eq,
                 bounds=bounds, method="highs")
    if not res.success:
        raise RuntimeError(res.message)
    strategy = res.x[:m]
    value = res.x[-1]
    return strategy, value
x_star, v_A = defensive_row_player(A)
y_star, w_B = defensive_row_player(B.T)
print("Захисна стратегія гравця А:", np.round(x_star, 3))
print("Гарантований результат гравця А:", round(v_A, 3))
print("Захисна стратегія гравця В:", np.round(y_star, 3))
print("Гарантований результат гравця В:", round(w_B, 3))

```

Додаток 2. Розв'язання робастної оптимізації для прямокутної області невизначеності.

```
import numpy as np
from scipy.optimize import linprog
A = np.array([
    [-150,-150,-150,-150,-150,-150,-150,-150,-150],
    [-54,-48,-76,17,-27,5,-59,-14,-46,19],
    [101,29,23,21,141,60,99,19,108,60],
    [128,121,67,62,83,190,98,127,82,159],
    [101,166,140,105,96,132,200,140,140,157],
    [249,200,251,250,208,172,226,275,279,233],
    [339,366,348,392,434,350,272,352,431,487],
    [398,468,432,466,511,589,450,347,449,589],
    [372,339,418,353,421,462,541,381,307,394],
    [148,244,186,238,190,262,278,337,198,192]
])
B = np.array([
    [-150,-67,92,171,237,458,419,268,159,2],
    [-150,-60,22,166,322,419,454,335,113,87],
    [-150,-91,17,103,301,502,444,297,181,19],
    [-150,-4,13,93,233,509,506,330,108,62],
    [-150,-50,130,122,208,413,557,364,172,10],
    [-150,-16,49,243,268,351,450,443,180,75],
    [-150,-81,89,152,381,443,361,338,255,65],
    [-150,-30,14,182,327,566,462,253,157,119],
    [-150,-62,104,139,312,578,579,338,115,27],
    [-150,4,57,217,331,508,641,445,163,44]
])
```

```

pho = 0.10
A_box = A - pho * np.abs(A)
B_box = B - pho * np.abs(B)
def security_strategy_row_player(P):
    n, m = P.shape
    c = np.zeros(n + 1); c[-1] = -1
    A_ub, b_ub = [], []
    for j in range(m):
        row = np.zeros(n + 1)
        row[:n] = -P[:, j]
        row[-1] = 1
        A_ub.append(row); b_ub.append(0)
    A_eq = [np.append(np.ones(n), 0)]
    b_eq = [1]
    bounds = [(0, None)] * n + [(None, None)]
    res = linprog(c, A_ub, b_ub, A_eq, b_eq, bounds, method="highs")
    return res.x[:n], res.x[-1]
def security_strategy_col_player(P):
    n, m = P.shape
    c = np.zeros(m + 1); c[-1] = -1
    A_ub, b_ub = [], []
    for i in range(n):
        row = np.zeros(m + 1)
        row[:m] = -P[i, :]
        row[-1] = 1
        A_ub.append(row); b_ub.append(0)
    A_eq = [np.append(np.ones(m), 0)]
    b_eq = [1]
    bounds = [(0, None)] * m + [(None, None)]
    res = linprog(c, A_ub, b_ub, A_eq, b_eq, bounds, method="highs")

```

```
    return res.x[:m], res.x[-1]
x_box, v_box = security_strategy_row_player(A_box)
y_box, w_box = security_strategy_col_player(B_box)
print("Робастна прямокутна стратегія гравця А:", np.round(x_box, 3))
print("Гарантований результат А:", round(v_box, 3))
print("Робастна прямокутна стратегія гравця В:", np.round(y_box, 3))
print("Гарантований результат В:", round(w_box, 3))
```

Додаток 3. Розв'язання робастної оптимізації для еліпсоїдальної області невизначеності.

```
import numpy as np
from scipy.optimize import minimize
A = np.array([
    [-150,-150,-150,-150,-150,-150,-150,-150,-150],
    [-54,-48,-76,17,-27,5,-59,-14,-46,19],
    [101,29,23,21,141,60,99,19,108,60],
    [128,121,67,62,83,190,98,127,82,159],
    [101,166,140,105,96,132,200,140,140,157],
    [249,200,251,250,208,172,226,275,279,233],
    [339,366,348,392,434,350,272,352,431,487],
    [398,468,432,466,511,589,450,347,449,589],
    [372,339,418,353,421,462,541,381,307,394],
    [148,244,186,238,190,262,278,337,198,192]
])
B = np.array([
    [-150,-67,92,171,237,458,419,268,159,2],
    [-150,-60,22,166,322,419,454,335,113,87],
    [-150,-91,17,103,301,502,444,297,181,19],
    [-150,-4,13,93,233,509,506,330,108,62],
    [-150,-50,130,122,208,413,557,364,172,10],
    [-150,-16,49,243,268,351,450,443,180,75],
    [-150,-81,89,152,381,443,361,338,255,65],
    [-150,-30,14,182,327,566,462,253,157,119],
    [-150,-62,104,139,312,578,579,338,115,27],
    [-150,4,57,217,331,508,641,445,163,44]
])
```

```

eps = 0.10
def robust_player_A(A, eps):
    n, m = A.shape
    def obj(z):
        return -z[-1]
    cons = []
    cons.append({
        'type': 'eq',
        'fun': lambda z: np.sum(z[:n]) - 1.0
    })
    for j in range(m):
        def ineq_fun(z, j=j):
            x = z[:n]
            v = z[-1]
            nom = A[:, j] @ x
            rad = eps * np.linalg.norm(np.abs(A[:, j]) * x, 2)
            return nom - rad - v
        cons.append({'type': 'ineq', 'fun': ineq_fun})

    bounds = [(0.0, 1.0)] * n + [(None, None)]
    z0 = np.concatenate([np.ones(n) / n, np.array([0.0])])
    res = minimize(obj, z0, method='SLSQP', bounds=bounds, constraints=cons)
    x_star = res.x[:n]
    v_star = res.x[-1]
    return x_star, v_star, res
def robust_player_B(B, eps):
    n, m = B.shape
    def obj(z):
        return -z[-1]
    cons = []

```

```

cons.append({
    'type': 'eq',
    'fun': lambda z: np.sum(z[:m]) - 1.0
})
for i in range(n):
    def ineq_fun(z, i=i):
        y = z[:m]
        w = z[-1]
        nom = B[i, :] @ y
        rad = eps * np.linalg.norm(np.abs(B[i, :]) * y, 2)
        return nom - rad - w
    cons.append({'type': 'ineq', 'fun': ineq_fun})
bounds = [(0.0, 1.0)] * m + [(None, None)]
z0 = np.concatenate([np.ones(m) / m, np.array([0.0])])
res = minimize(obj, z0, method='SLSQP', bounds=bounds, constraints=cons)
y_star = res.x[:m]
w_star = res.x[-1]
return y_star, w_star, res
if __name__ == "__main__":
    x_ell, v_ell, resA = robust_player_A(A, eps)
    y_ell, w_ell, resB = robust_player_B(B, eps)
    print("Робастна еліпсоїдна стратегія гравця А:", np.round(x_ell, 3))
    print("Гарантований результат А:", round(v_ell, 3))
    print("Робастна еліпсоїдна стратегія гравця В:", np.round(y_ell, 3))
    print("Гарантований результат В:", round(w_ell, 3))

```

Додаток 4. Обчислення перетину прямокутної та еліпсоїдальної невизначеності робастної оптимізації біматричної гри.

```
import numpy as np
from scipy.optimize import minimize

A = np.array([
    [-150,-150,-150,-150,-150,-150,-150,-150,-150],
    [-54,-48,-76,17,-27,5,-59,-14,-46,19],
    [101,29,23,21,141,60,99,19,108,60],
    [128,121,67,62,83,190,98,127,82,159],
    [101,166,140,105,96,132,200,140,140,157],
    [249,200,251,250,208,172,226,275,279,233],
    [339,366,348,392,434,350,272,352,431,487],
    [398,468,432,466,511,589,450,347,449,589],
    [372,339,418,353,421,462,541,381,307,394],
    [148,244,186,238,190,262,278,337,198,192]
])

B = np.array([
    [-150, -67, 92, 171, 237, 458, 419, 268, 159, 2],
    [-150, -60, 22, 166, 322, 419, 454, 335, 113, 87],
    [-150, -91, 17, 103, 301, 502, 444, 297, 181, 19],
    [-150, -4, 13, 93, 233, 509, 506, 330, 108, 62],
    [-150, -50, 130, 122, 208, 413, 557, 364, 172, 10],
    [-150, -16, 49, 243, 268, 351, 450, 443, 180, 75],
    [-150, -81, 89, 152, 381, 443, 361, 338, 255, 65],
    [-150, -30, 14, 182, 327, 566, 462, 253, 157, 119],
    [-150, -62, 104, 139, 312, 578, 579, 338, 115, 27],
    [-150, 4, 57, 217, 331, 508, 641, 445, 163, 44]
])
```

```

nA, mA = A.shape
rho = 1.0
eps = 3.1
A_hat = 0.1 * np.abs(A)
n_u = nA * mA
n_z = nA * mA
def objectiveA(vars):
    return -vars[-1]
def extractA(vars):
    x = vars[:nA]
    u_flat = vars[nA:nA+n_u]
    z_flat = vars[nA+n_u:nA+n_u+n_z]
    U = u_flat.reshape((nA, mA))
    Z = z_flat.reshape((nA, mA))
    v = vars[-1]
    return x, U, Z, v
constraintsA = []
constraintsA.append({'type': 'eq', 'fun': lambda vars: np.sum(vars[:nA]) - 1.0})

for i in range(nA):
    for j in range(mA):
        u_idx = nA + i*mA + j
        z_idx = nA + n_u + i*mA + j
        def make_upper(i_f, u_f, z_f):
            return lambda vars: vars[i_f] - vars[z_f] - vars[u_f]
        def make_lower(i_f, u_f, z_f):
            return lambda vars: -vars[i_f] + vars[z_f] - vars[u_f]
        constraintsA.append({'type': 'ineq', 'fun': make_upper(i, u_idx, z_idx)})
        constraintsA.append({'type': 'ineq', 'fun': make_lower(i, u_idx, z_idx)})
    for j in range(mA):

```

```

def make_combined(j_f):
    def constraint(vars):
        x, U, Z, v = extractA(vars)
        nominal = A[:, j_f] @ x
        box_penalty = rho * np.sum(A_hat[:, j_f] * U[:, j_f])
        ell_penalty = eps * np.linalg.norm(A_hat[:, j_f] * Z[:, j_f], 2)
        return nominal - box_penalty - ell_penalty - v
    return constraint

constraintsA.append({'type': 'ineq', 'fun': make_combined(j)})

boundsA = [(0.0, 1.0)] * nA + [(0.0, None)] * n_u + [(None, None)] * n_z + [(None,
None)]
x0 = np.ones(nA) / nA
u0 = np.zeros(n_u)
z0 = np.repeat(x0, mA)
v0 = 0.0
vars0A = np.concatenate([x0, u0, z0, [v0]])
C = B.T
nB, mB = C.shape
C_hat = 0.1 * np.abs(C)

n_u2 = nB * mB
n_z2 = nB * mB
def objectiveB(vars):
    return -vars[-1]
def extractB(vars):
    y = vars[:nB]
    u_flat = vars[nB:nB+n_u2]
    z_flat = vars[nB+n_u2:nB+n_u2+n_z2]
    U = u_flat.reshape((nB, mB))
    Z = z_flat.reshape((nB, mB))

```

```

w = vars[-1]
return y, U, Z, w
constraintsB = []
constraintsB.append({'type': 'eq', 'fun': lambda vars: np.sum(vars[:nB]) - 1.0})
for i in range(nB):
    for j in range(mB):
        u_idx = nB + i*mB + j
        z_idx = nB + n_u2 + i*mB + j
        def make_upper(i_f, u_f, z_f):
            return lambda vars: vars[i_f] - vars[z_f] - vars[u_f]
        def make_lower(i_f, u_f, z_f):
            return lambda vars: -vars[i_f] + vars[z_f] - vars[u_f]
        constraintsB.append({'type': 'ineq', 'fun': make_upper(i, u_idx, z_idx)})
        constraintsB.append({'type': 'ineq', 'fun': make_lower(i, u_idx, z_idx)})
    for j in range(mB):
        def make_combined(j_f):
            def constraint(vars):
                y, U, Z, w = extractB(vars)
                nominal = C[:, j_f] @ y
                box_penalty = rho * np.sum(C_hat[:, j_f] * U[:, j_f])
                ell_penalty = eps * np.linalg.norm(C_hat[:, j_f] * Z[:, j_f], 2)
                return nominal - box_penalty - ell_penalty - w
            return constraint
        constraintsB.append({'type': 'ineq', 'fun': make_combined(j)})
boundsB = [(0.0, 1.0)] * nB + [(0.0, None)] * n_u2 + [(None, None)] * n_z2 +
[(None, None)]
y0 = np.ones(nB) / nB
u0 = np.zeros(n_u2)
z0 = np.repeat(y0, mB)
w0 = 0.0

```

```

vars0B = np.concatenate([y0, u0, z0, [w0]])
print("Розв'язування для гравця А")
resultA = minimize(objectiveA, vars0A, method='SLSQP', bounds=boundsA,
                   constraints=constraintsA, options={'maxiter': 1000})
x_opt, U_opt, Z_opt, v_opt = extractA(resultA.x)
print("\nУспіх:", resultA.success)
print("Гарантований результат А =", round(v_opt, 3))
print("Стратегія гравця А =", np.round(x_opt, 3))
print("Розв'язування для гравця В")
resultB = minimize(objectiveB, vars0B, method='SLSQP',
                   bounds=boundsB, constraints=constraintsB,
                   options={'maxiter': 1000})
y_opt, U_opt, Z_opt, w_opt = extractB(resultB.x)
print("\nУспіх:", resultB.success)
print("Гарантований результат В =", round(w_opt, 3))
print("Стратегія гравця В =", np.round(y_opt, 3))

```