МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Л. М. Малярець
К. О. Ковальова

ДОСЛІДЖЕННЯ ОПЕРАЦІЙ ТА МЕТОДИ ОПТИМІЗАЦІЇ

Лабораторний практикум в середовищі MATLAB

Харків
ХНЕУ ім. С. Кузнеця
2018

УДК 519.8(075.034)
М21

Авторський колектив: д-р екон. наук, професор Л. М. Малярець – лабораторні роботи 5 – 8; канд. техн. наук, доцент К. О. Ковальова – лабораторні роботи 1 – 4.

Рецензенти: д-р фіз.-мат. наук, професор кафедри гігієни і соціальної медицини Харківського національного університету імені В. Н. Каразіна *О. В. Мартиненко*; завідувач кафедри математики і математичних методів в економіці Донецького національного університету імені Василя Стуса, д-р екон. наук, професор *В. В. Христіановський*; проректор з підготовки наукових кадрів Східноєвропейського університету економіки і менеджменту, д-р екон. наук, професор *Г. О. Ус*.

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 7 від 26.03.2018 р.

Самостійне електронне текстове мережеве видання

Малярець Л. М.

М21 Дослідження операцій та методи оптимізації : лабораторний практикум в середовищі MATLAB [Електронний ресурс] / Л. М. Малярець, К. О. Ковальова. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 123 с.

ISBN 978-966-676-738-0

Подано матеріал для виконання лабораторних робіт із навчальної дисципліни, стисло викладено основні теоретичні відомості щодо вбудованих функцій середовища MATLAB, що використовуються за відповідною темою, наведено приклади розв'язання типових задач, варіанти лабораторних робіт.

Рекомендовано для використання студентами у процесі неперервної математичної підготовки, магістрам, аспірантам для проведення наукових досліджень на основі побудови оптимізаційних моделей у середовищі MATLAB, економістам-практикам для обґрунтування та ухвалення управлінських рішень із використанням комп'ютерних програм.

УДК 519.8(075.034)

ISBN 978-966-676-738-0

Вступ

Сучасному етапу розвитку економіки притаманний високий рівень її формалізації. В умовах стрімкого зростання значення аналітичних досліджень в управлінні соціально-економічними процесами майбутнім економістам потрібна ґрунтовна математична підготовка, що давала б можливість застосовувати математичний інструментарій до розв'язання широкого кола проблем у сфері їх професійної діяльності. Економіко-математичні методи є тим інструментом дослідження економічних систем і процесів різної складності, що дозволяє отримувати достовірну інформацію щодо характеристик економічних процесів та явищ. Саме за допомогою математичних методів розробляються економіко-математичні моделі економічних процесів, які в подальшому є підґрунтям формування управлінських рішень щодо оптимізації цих процесів під час розв'язання реальних аналітичних задач.

Завдяки широкому впровадженню інформаційних технологій та комп'ютеризації у всіх сферах людської діяльності взагалі, а також у процесі отримання освіти зокрема залишається у минулому таке уявлення про навчання, як проста передача інформації від викладача до студента. Упроваджується концептуально новий підхід до процесу навчання, що передбачає спрямування на кінцевий результат, а саме, на формування компетентного фахівця, який не тільки здобув певні знання, але й отримав необхідні вміння і навички, які дозволяють ефективно використовувати набуті знання в професійній діяльності.

"Дослідження операцій та методи оптимізації" – це комплексна економіко-математична дисципліна, що займається побудовою, аналізом і застосуванням математичних моделей прийняття оптимальних рішень під час проведення операцій. Тут під операцією слід розуміти систему керованих дій, об'єднану єдиним задумом і спрямовану на досягнення певної мети, а під математичною моделлю – сукупність співвідношень: рівнянь, нерівностей, логічних умов, операторів і так далі, що визначають характеристики стану об'єкта моделювання, параметри функціонування і розвитку. Однією з умов досконалого вивчення цієї дисципліни є набуття фахівцем знань в області комп'ютерного моделювання за допомогою середовища MATLAB.

До складу програмного середовища MATLAB входить оптимізаційний тулбокс *Optimization Toolbox*, який включає програми широко відомих методів мінімізації і максимізації лінійних і нелінійних функцій. Ці програми можуть бути використані для розв'язання складних задач оптимізації вартості, надійності

та якості для різних додатків. У пакет включені версії традиційних і новітніх алгоритмів оптимізації, в тому числі безумовна оптимізація (метод симплексного пошуку Нелдера-Міда і квазі-ньютонівський метод БФШ), умовна, багатокритеріальна оптимізація і метод мінімакса, методи лінійного та квадратичного програмування.

Детальний приклад того, як використовувати *Optimization Toolbox* наведено у цьому лабораторному практикумі.

Метою поданого матеріалу є висвітлення виконання лабораторних робіт з дисципліни "Дослідження операцій та методи оптимізації" з використанням програмного продукту MATLAB для формування у студентів системи знань з методології застосування математичного інструментарію для побудови та використання різних типів оптимізаційних моделей, набуття необхідної сукупності теоретичних і практичних знань для розв'язання конкретних завдань, які постають у процесі побудови економіко-математичних моделей на сучасному етапі розвитку.

У процесі вивчення дисципліни "Дослідження операцій та методи оптимізації" з використанням середовища MATLAB відбувається формування у студентів таких професійних компетентностей:

аналіз й обробка даних з використанням вбудованих функцій середовища MATLAB, необхідних для розв'язання поставлених оптимізаційних задач;

використання вбудованих функцій тулбоксу *Optimization Toolbox* середовища MATLAB для обробки економічних даних відповідно до поставленої задачі, аналіз результатів математичного моделювання й обґрунтування отриманих висновків.

У результаті вивчення дисципліни студент набуває: знання, вміння та навички щодо побудови математичних моделей та використання методів лінійного програмування для розв'язання оптимізаційних задач в економіці, економічних задач за допомогою квадратичного програмування, методів дробово-лінійного програмування, теорії ігор тощо; знання, вміння та навички щодо застосування методів управління запасами та теорії масового обслуговування.

Лабораторний практикум складено з урахуванням робочої програми викладання дисципліни "Дослідження операцій та методи оптимізації", охоплює всі теми курсу та складається з восьми лабораторних робіт.

Лабораторна робота 1
Задача лінійного програмування та методи її розв'язування: графічне розв'язування ЗЛП

Мета роботи: пояснити геометричну інтерпретацію задач лінійного програмування (ЗЛП), вивчити графічний метод розв'язування ЗЛП, його можливості та область застосування за допомогою вбудованих функцій середовища MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель вихідної задачі.

2. За допомогою вбудованої функції `linprog` середовища MATLAB знайти її оптимальний розв'язок.

3. Засобами середовища MATLAB відобразити графічно отриманий розв'язок.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

1.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

LINPROG **Розв'язання задачі лінійного програмування**

Необхідно мінімізувати значення виразу $f^T x$, тобто знайти $\min_x f^T x$ за обмежень:

$$A \cdot x \le b;$$
$$Aeq \cdot x = beq;$$
$$lb \le x \le ub,$$

де f, x, b, lb та ub – вектори, а A та Aeq – матриці.

Синтаксис

```
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
```

```
[x,fval] = linprog(...)
[x,fval,exitflag] = linprog(...)
[x,fval,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)
```

Опис

`Linprog` розв'язує задачу лінійного програмування.

`x = linprog (f, A, b)` знаходить $\min_x f^T x$ за умови, що $A \cdot x \le b$.

`x = linprog (f, A, b, Aeq, beq)` розв'язує зазначену задачу за умови додаткового виконання обмежень у вигляді рівностей $Aeq \cdot x = beq$. Якщо немає нерівностей, то встановлюється `A = []` та `b = []`.

`x = linprog (f, A, b, Aeq, beq, lb, ub)` визначає набір нижніх і верхніх меж для проектованих змінних `x`, так що розв'язок завжди знаходиться в діапазоні $lb \le x \le ub$. Якщо немає нерівностей, то встановлюється `Aeq = []` та `beq = []`.

`x = linprog (f, A, b, Aeq, beq, lb, ub, x0)` встановлює початкову точку як `x0`. Ця опція має місце тільки для середньомасштабного алгоритму (`options.LargeScale` дорівнює `'off'`). Крупномасштабний алгоритм, що приймається за замовчуванням, ігнорує будь-яку стартову точку.

`x = linprog (f, A, b, Aeq, beq, lb, ub, x0, options)` проводить оптимізацію з певними у структурній опції параметрами оптимізації.

`[X, fval] = linprog (...)` повертає значення цільової функції `fun` як розв'язок від `x`: `fval = f'*x`.

`[X, lambda, exitflag] = linprog (...)` повертає значення `exitflag`, яке містить опис вихідних умов.

`[X, lambda, exitflag, output] = linprog (...)` повертає структурний вихід з інформацією про оптимізацію.

`[X, fval, exitflag, output, lambda] = linprog (...)` повертає структурну `lambda`, чиї поля включають множники Лагранжа як розв'язок від `x`.

Примітка: якщо якийсь із вхідних параметрів відсутній, на його місце необхідно поставити квадратні дужки [], за винятком випадку, коли це останній параметр у списку.

Аргументи

Табл. 1.1 містить загальний опис аргументів, переданих в `linprog`. У табл. 1.1 міститься опис функціонально-специфічних деталей для параметрів `options`.

Таблиця 1.1

Вхідні аргументи функції `linprog`

Аргументи	Опис
`A, b`	Матриця `A` і вектор `b` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних нерівностей і вектора правої частини: `A*x <= b`
`Aeq, beq`	Матриця `Aeq` і вектор `beq` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних рівностей і вектора правої частини: `Aeq*x = beq`
`f`	Вектор коефіцієнтів для лінійного члена в лінійному рівнянні: `f'*x`
`lb, ub`	Нижній і верхній обмежувальні вектори (або матриці). Зазвичай ці аргументи мають розмірність типу `x`. Однак, якщо в `lb` є дещо менше елементів, ніж `x`, (наприклад, тільки `m`), то тільки перші `m` елементів в `x` мають межу знизу; верхня межа `ub` може визначатися таким саме чином. Також за допомогою `-Inf` (для нижніх меж) або `Inf` (для верхніх меж) можна визначити так звані змінні без обмежень. Наприклад, якщо `lb(i) = -Inf` то змінна `x(i)` є необмеженою знизу
`options`	Структура з параметрами опцій оптимізації, за допомогою якої задаються параметри опцій оптимізації. Більш детально інформацію про ці параметри можна знайти в табл. А. 1 додатка А
`x0`	Стартова точка (скаляр, вектор або матриця)

Табл. 1.2 містить загальний опис аргументів, що повертаються функцією `linprog`. У цій таблиці наводяться загальні специфічні деталі для величин `exitflag`, `lambda` і `output`.

Функція `linprog` може використовувати алгоритм великої розмірності lipsol або алгоритм середньої розмірності (метод проекцій).

Розглянемо приклад вирішення задачі лінійного програмування у середовищі MATLAB.

Таблиця 1.2

Вихідні аргументи функції `linprog`

Аргументи	Опис
`exitflag`	Описує вихідні умови. • `> 0` Ця функція збігається до розв'язку `x` • `0` Максимальне число оцінки функції або ітерації було перевищено • `<0` Функція не збігається до деякого розв'язку
`lambda`	Структура, яка містить множники Лагранжа у разі розв'язання за `x` (розділеному за типами умов). Поле структури: • `lower` Нижні межі `lb` • `upper` Верхні межі `ub` • `ineqlin` Лінійні нерівності • `eqlin` Лінійні рівності
`output`	Структура, яка містить інформацію про оптимізацію. Поле структури: • `iterations` Число виконаних ітерацій • `algorithm` Алгоритм, що використовувався для розв'язання • `cgiterations` Число `PCG` ітерацій (тільки для крупно-масштабного алгоритму)

1.2. Розв'язування типових задач у середовищі MATLAB

Приклад 1.1. Компанія виробляє навантажувачі та візки. Від одного навантажувача компанія отримує дохід у розмірі 80 ум. од. і від одного візка – в розмірі 40 ум. од. Компанія також має три обробних центра, на яких виконуються операції металообробки, зварювання і збирання, необхідні для виробництва будь-якого з продуктів. Для інтервалу планування, що дорівнює місяцю, задана гранична виробнича потужність кожного обробного центру в годинах, а також кількість годин, необхідна на цьому центрі для виробництва одного навантажувача та візка. Ця інформація наведена в табл. 1.3.

Таблиця 1.3

Вихідні дані задачі

Виріб / Центр	Навантажувач (год. на од.)	Візок (год. на од.)	Загальна потужність (год.)
Металообробка	6	4	2400
Зварювання	2	3	1500
Збирання	9	3	2700

Потрібно скласти допустимий план робіт на місяць з максимальним прибутком.

Розв'язання. Математична модель задачі може бути записана таким чином:

$$f = 80x_1 + 40x_2 \to \max;$$
$$\begin{cases} 6x_1 + 4x_2 \le 2400 \\ 2x_1 + 3x_2 \le 1500 \\ 9x_1 + 3x_2 \le 2700; \end{cases}$$
$$x_1 \ge 0,\ x_2 \ge 0,$$

де x_1, x_2 – кількість вироблених навантажувачів і візків відповідно.

Представимо нашу задачу відповідно до заданої форми роботи команди:

$f = 80x_1 + 40x_2 \to \max$

Вектор коефіцієнтів цільової функції:
`f=[80; 40];`

$$\begin{cases} 6x_1 + 4x_2 \le 2400 \\ 2x_1 + 3x_2 \le 1500 \\ 9x_1 + 3x_2 \le 2700 \end{cases}$$

Матриця коефіцієнтів системи лінійних нерівностей:
`A=[6 4; 2 3; 9 3];`
Вектор вільних членів системи лінійних нерівностей:
`b=[2400; 1500; 2700];`

$x_1 \ge 0$
$x_2 \ge 0$

Вектор нижніх меж для змінних x_1, x_2:
`lb=zeros(2,1);`

Примітка: функція `linprog` шукає мінімум, а за умовою задачі необхідно знайти максимум. Тому, як один з варіантів, необхідно поставити знак мінус перед ім'ям вектора `f` для звернення до функції `linprog`.

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `linprog`, наведені у відповідній програмі (m-файл), який виглядає так:

```
clear all
close all
clc % Видаляються всі поточні змінні з пам'яті MATLAB,
    % закриваються всі графічні вікна, очищається екран
    % консолі;
f=[-80; -40]; % Вектор коефіцієнтів цільової функції;
```

```
%Матриця коефіцієнтів системи лінійних нерівностей:
A=[6 4; 2 3; 9 3];
b=[2400; 1500; 2700]; % Вектор вільних членів;
lb=zeros(2,1); % Нижні обмеження змінних;
% Звернення до програми лінійного програмування:
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);
% Вивід розрахункових даних
% Вивід оптимального рішення:
x
f = -fval
% Вивід додаткових параметрів оптимізації
% інформація про те, як завершилося розв'язання задачі:
exitflag
iter = output.iterations % кількість ітерацій;
algorithm = output.algorithm % алгоритм, що використовується;
ineqlin = lambda.ineqlin % активні обмеження у розв'язанні
нерівностей системи обмежень;
low = lambda.lower % активні обмеження за умови невід'ємності
змінних невід'ємності змінних;
% Графічне розв'язання задачі:
xs = [0:1:400];
y1 = max((2400-6.*xs)/4, 0);
y2 = max((1500-2.*xs)/3, 0);
y3 = max((2700-9.*xs)/3, 0);
plot(xs,y1,xs,y2,xs,y3)
grid on
legend('6x1+4x2<=2400','2x1+3x2<=1500','9x1+3x2<=2700')
xlabel('x1')
ylabel('x2')
hold on
ytop = min([y1; y2; y3]);
area(xs, ytop, 'FaceColor', [0.5 0.5 0.5]);
hold on
plot(x(1),x(2),'go')
```

Результати роботи можна подати таким чином:

```
Optimization terminated.

x =
  200.0000
  300.0000

f = 2.8000e+004
```

```
exitflag = 1

iter = 4

algorithm = large-scale: interior point

ineqlin =
    6.6667
    0.0000
    4.4444

low =
  1.0e-011 *
    0.3022
    0.0849
```

Таким чином, максимальний прибуток компанії склав 28 000 ум. од. від виробництва 200 навантажувачів і 300 візків.

На рис. 1.1 подано графічну інтерпретацію розв'язку поставленої задачі, отриману в результаті виконання наведених команд.

Отриманий результат також досить детально характеризує ітераційний процес. У програмі був використаний алгоритм крупномасштабної оптимізації (`algorithm = large-scale: interior point`), який успішно завершив роботу (`exitflag = 1`) за чотири ітерації (`iter = 4`). Ненульові елементи векторів у полях `lambda` вказують на активні обмеження у розв'язанні.

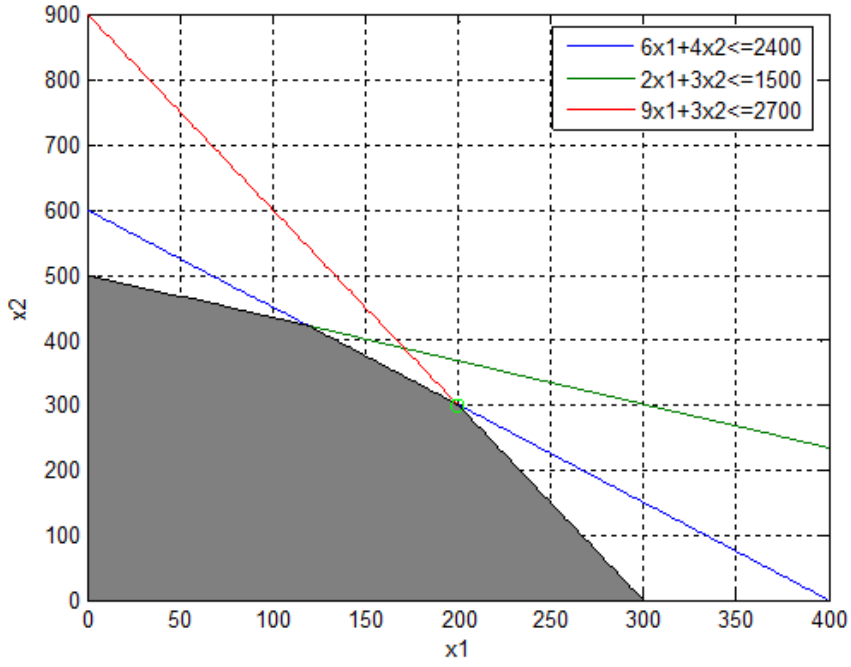

Рис. 1.1. **Область допустимих розв'язків задачі**

У нашому випадку перше та третє обмеження у вигляді нерівностей (в `lambda.ineqlin`) і обидва обмеження по знаку (в `lambda.lower`) є активними обмеженнями (тобто розв'язання знаходиться на обмежувальних умовах).

Запитання для самоперевірки

1. Які обов'язкові складові присутні у математичній моделі задачі лінійного програмування (ЛП)?

2. Наведіть основні теореми, на яких базується графічний метод розв'язання задач ЛП.

3. Наведіть алгоритм графічного методу розв'язання задач ЛП.

4. Яке призначення функції `linprog`?

5. Назвіть вхідні та вихідні параметри функції `linprog`.

6. Як перетворювати цільову функцію, максимум якої необхідно визначити, якщо функція `linprog` шукає лише мінімум задач ЛП?

7. Які функції *MATLAB* використовуються для графічної інтерпретації задач ЛП?

Завдання до лабораторної роботи

Для наведених задач ЛП необхідно:

1) побудувати математичну модель задачі;

2) з використанням вбудованої функції `linprog` знайти її оптимальний розв'язок;

3) зробити висновки щодо отриманих результатів;

4) зобразити графічно засобами MATLAB область допустимих розв'язків задачі та позначити на ній оптимальний розв'язок.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Задача 1.1. Для виробництва комп'ютерних столів I і II видів потрібні три типи ресурсів: дерево, пластик і трудовитрати. Потреби в ресурсах для виробництва одного столу кожного виду, запаси ресурсів, а також прибуток від реалізації одного столу кожного виду задані в табл. 1.4.

Таблиця 1.4

Вихідні дані

Типи ресурсів	Одиниця продукції виду I	Одиниця продукції виду II	Запас ресурсу
Дерево ($м^2$)	1	3	24
Пластик ($м^2$)	4	1	24
Трудовитрати (люд/год)	3	2	23
Прибуток (грн)	200	300	

Потрібно знайти план випуску продукції, що дозволяє отримати найбільший прибуток.

Задача 1.2. Підприємство має можливість придбати не більше 19 тритонних автомашин і не більше 17 п'ятитонних. Відпускна ціна тритонної вантажівки – 4 000 ум. од., п'ятитонної – 5 000 ум. од. Підприємство може виділити для придбання автомашин 141 тис. ум. од. Скільки потрібно придбати автомашин, щоб їх сумарна вантажопідйомність була максимальною?

Задача 1.3. Експериментальна лабораторія хімічного заводу розробила два нових види реактивів, які мають величезне значення в господарській діяльності регіону. Прибуток від продажу однієї тонни реактиву, відповідно, становить 80 і 10 ум. од. Але під час виробництва цих реактивів в атмосферу виділяються шкідливі речовини A, B, C, D, E. Норми викидів речовини A в місяць складають 5 мл, B – 8 мл, C – 9 мл, D – 10 мл, E – 10 мл. Викиди цих речовин у процесі виробництва однієї тонни кожного реактиву наведені в табл. 1.5.

Таблиця 1.5

Вихідні дані

Шкідливі речовини	Вид реактиву	
	I	II
A, мл	2	1
B, мл	0	4
C, мл	8	1
D, мл	4	6
E, мл	5	0

Необхідно визначити, в якій кількості дані реактиви можна виробляти, щоб не перевищити норми викидів шкідливих речовин і отримати максимальний прибуток.

Задача 1.4. Процес виготовлення глюкози та патоки на крохмально-паточному комбінаті є виконанням таких операцій, як розщеплення сировини, вичавлювання і варення. Час роботи обладнання у ході виконання кожної операції обмежений та складає, відповідно, 16, 12 і 14 годин. Норми часу обробки продукції для кожної операції, а також прибуток, отримуваний комбінатом від реалізації одиниці продукції, наведені в табл. 1.6.

Таблиця 1.6

Вихідні дані

Операції	Норми витрат часу на виготовлення одиниці продукції, год/кг	
	Патока	Глюкоза
Розщеплення	4	4
Вичавлювання	3	4
Варення	7	2
Прибуток від одиниці продукції, ум. од./кг	3	6

Необхідно визначити план випуску продукції, що забезпечує комбінату максимальний прибуток.

Задача 1.5. Підприємство випускає два види продукції: калоші та валянки. На виготовлення однієї пари калош потрібно затратити 2 кг бавовни, 3 кг вовни, 5 кг гуми. На виготовлення однієї пари валянок потрібно затратити 5 кг бавовни, 4 кг вовни, 3 кг гуми. Виробництво забезпечене сировиною кожного типу в кількості 432 кг, 424 кг, 582 кг, відповідно. Ринкова ціна однієї пари калош становить 34 тис. ум. од., а валянок – 50 тис. ум. од. Складіть план виробництва виробів, що забезпечує максимальну виручку від їх реалізації.

Задача 1.6. Кондитерська фабрика для виробництва двох видів карамелі А і В використовує три види основної сировини: цукровий пісок, патоку і фруктове пюре. Норми витрати сировини кожного виду на виробництво 1 т карамелі даного виду, загальна кількість сировини кожного виду, а також прибуток від реалізації 1 т карамелі даного виду наведені в табл. 1.7.

Таблиця 1.7

Вихідні дані

Вид сировини	Норми витрати сировини (т) на 1 т карамелі		Загальна кількість сировини (т)
	Карамель *A*	Карамель *B*	
Цукровий пісок	8	5	80
Патока	4	4	60
Фруктове пюре	0	1	12
Прибуток від реалізації 1 т карамелі (ум. од)	45	58	

Знайдіть план виробництва карамелі, що забезпечує максимальний прибуток від її реалізації.

Задача 1.7. Магазин здійснює реалізацію товарів двох видів – головні убори та рукавиці. Дані про нормовитрати ресурсів, їх запаси та про прибуток від реалізації одиниці товару подані в табл. 1.8.

Таблиця 1.8

Вихідні дані

Види ресурсів	Витрати ресурсів на одиницю товару		Обсяг наявних ресурсів
	Головні убори	Рукавиці	
Матеріальні ресурси, грн	4	3	24
Трудові ресурси, люд.-хв.	7	8	56
Прибуток від реалізації одиниці товару, грн	0,5	0,6	

Визначте структуру товарообігу виходячи з умови отримання максимального прибутку.

Задача 1.8. Для виробництва столів і шаф меблева фабрика використовує ресурси деревини двох видів. Норми витрат цих ресурсів, їх загальна кількість, а також дохід від реалізації одного виробу наведені в табл. 1.9.

Таблиця 1.9

Вихідні дані

Ресурси деревини	Норми витрат ресурсів на один виріб, $м^3$ / виріб		Загальна кількість ресурсів, $м^3$
	Столи	Шафи	
1 виду	3	2	30
2 виду	1	4	50
Дохід від реалізації одного виробу грн / виріб	50	80	

Визначте, скільки столів і шаф повинна виготовити фабрика, щоб забезпечити максимальний прибуток.

Задача 1.9. На двох типах технологічного обладнання підприємство може виробляти два види виробів, для кожного з яких задані витрати часу на одиницю продукції. Фонд часу за групами устаткування, а також прибуток від випуску одного виробу наведені в табл. 1.10

Таблиця 1.10

Вихідні дані

Тип обладнання	Витрати часу на виробництво одного виробу, год. / вироб.		Фонд часу, год.
	1 вид	2 вид	
1 тип	1	2	6
2 тип	4	4	8
Прибуток від одиниці продукції, грн. / вироб	1	2	

Визначте план випуску продукції, що забезпечує максимум прибутку, якщо перший вид виробу не може бути менший одиниці.

Задача 1.10. Для виробництва трюмо та тумбочок меблевий комбінат використовує деревину трьох видів. Запаси деревини, норми його витрат і прибуток від реалізації виробів наведені в табл. 1.11.

Таблиця 1.11

Вихідні дані

Види деревини	Норми витрат ресурсів, $м^3$ / виріб		Запаси деревини, $м^3$
	Трюмо	Тумбочка	
1 вид	4	3	50
2 вид	6	5	70
3 вид	1	1	32
4 вид	5	3	45
Прибуток від одиниці продукції	8	12	

Визначте оптимальний план випуску продукції, за якого прибуток від реалізації виробів буде максимальним.

Задача 1.11. У студентській їдальні для виготовлення бутербродів двох видів використовують чотири види ресурсів, загальні обсяги запасів яких і норми витрат вказані в табл. 1.12. Відомий також прибуток, отримуваний їдальнею від реалізації однієї партії бутербродів кожного виду.

Таблиця 1.12

Вихідні дані

Ресурси	Норми витрати ресурсів на одну партію бутербродів, кг / парт		Наявний обсяг ресурсів, кг
	I вид бутербродів	II вид бутербродів	
I вид	4	3	42
II вид	2	5	56
III вид	3	6	38
IV вид	5	7	40
Прибуток від реалізації однієї партії бутербродів, грн	50	70	

Заплануйте випуск партій бутербродів в такій кількості, щоб загальний прибуток їдальні був максимальний. Вам необхідно врахувати, що бутербродів першого виду необхідно виготовити не менше чотирьох партій.

Задача 1.12. Для відгодівлі тварин використовуються два продукти – П1 і П2, що містять білок, кальцій і вітаміни. Зміст цих поживних речовин у продуктах відгодівлі відомі (табл. 1.13).

Таблиця 1.13

Вихідні дані

Поживні речовини	Кількість одиниць поживних речовин в одному кг корму	
	П1	П2
Білок	10	8
Кальцій	12	6
Вітаміни	4	2
Ціна 1 кг продукту	30	20

Відомо, що для нормальної відгодівлі треба спожити не менше : 30 г білка, 18 г кальцію і 60 г вітамінів. Визначте оптимальний раціон годування тварин за умови мінімальної вартості.

Задача 1.13. Для нормального розвитку промислового рибництва в господарстві необхідно, щоб риба щодня отримувала чотири види поживних речовин в кількості, відповідно, 20, 15, 18 і 12 од. Ці поживні речовини містяться в двох видах кормів. Вміст поживних речовин в одному кг корму наведено в табл. 1.14.

Таблиця 1.14

Вихідні дані

Поживні речовини	Кількість одиниць поживних речовин в одному кг корму	
	I виду	II виду
A_1	10	8
A_2	12	6
A_3	4	2
A_4	5	7

Необхідно скласти оптимальний раціон годування риб, якщо відомо, що ціна одного кг I виду корму 2 грн, а II виду – 1 грн.

Задача 1.14. За рахунок меліоративних робіт площа ферми в господарстві зросла на 120 га. Цю площу було вирішено відвести під посів двох найбільш ефективних для господарства культур: проса та гречки. Вирощування культур характеризується показниками, поданими в табл. 1.15.

Таблиця 1.15

Вихідні дані

	Види зернових культур		Ресурси ферми
	Просо	Гречка	
Місце під зберігання зерна	3	6	40
Витрати на вирощування зернових	6	2	150
Прибуток від продажу, грн	143	60	

Побудуйте математичну модель оптимального вирощування зернових культур таким чином, щоб прибуток ферми був максимальним. Згідно з побудованою математичною моделлю визначте оптимальний план вирощування зернових і максимальний прибуток ферми.

Задача 1.15. Прядильна фабрика для виробництва двох видів пряжі використовують три типи сировини: чисту шерсть, капрон і акрил. У табл. 1.16 указані норми витрат сировини, її загальна кількість, яка може бути використана фабрикою протягом року, та прибуток від реалізації тонни пряжі кожного виду.

Таблиця 1.16

Вихідні дані

Тип сировини	Норми витрати сировини на 1 т пряжі (т)		Кількість сировини (т)
	Вид 1	Вид 2	
Шерсть	5	2	60
Капрон	1	6	62
Акрил	4	2	50
Прибуток від продажу, тис. грн	11	9	

Потрібно скласти оптимальний план виробництва пряжі з метою максимізації прибутку.

Задача 1.16. Процес виготовлення шкіряних курток і пальто передбачає проходження виробів через дубильний, розкрійний та пошивний цехи. Фонд часу роботи кожного з них, норми часу обробки виробів у кожному з цехів, а також прибуток, отримуваний підприємством від випуску одиниці продукції, наведені в табл. 1.17.

Таблиця 1.17

Вихідні дані

Цехи	Норми витрат часу на одиницю продукції, год/виріб		Фонд часу роботи цехів, год.
	Куртки	Пальто	
Дубильний	2	7	42
Розкрійний	3	5	30
Пошивний	8	4	32
Прибуток від одиниці продукції, грн/вироб	3	5	

Знайдіть план випуску виробів, що забезпечує підприємству максимальний прибуток.

Задача 1.17. Для виробництва столів і стільців меблевий комбінат використовує деревину трьох видів. Запаси деревини, норми її витрати, плановий асортимент продукції і її собівартість наведені в табл. 1.18.

Таблиця 1.18

Вихідні дані

Вид деревини	Норми витрат ресурсів, $м^3$ / виріб		Запаси деревини, $м^3$
	Столи	Стільці	
I вид	2	4	50
II вид	3	6	70
III вид	7	2	65
Плановий асортимент, шт.	5	8	
Прибуток від одиниці продукції, грн/виріб	5	3	

Визначте оптимальний план випуску продукції з метою максимізації прибутку меблевого комбінату виходячи з неможливості перевиконання планового асортименту.

Задача 1.18. Фабрика з розфасування чаю випускає чай сорту А та Б, змішуючи три інгредієнта: індійський, грузинський і краснодарський чаї. У табл. 1.19 наведено норми витрат інгредієнтів, обсяг запасів кожного з них і прибуток від реалізації 1 т чаю сортів А і Б.

Таблиця 1.19

Вихідні дані

Складові	Норми витрат		Обсяг запасів
	Сорт А	Сорт Б	
Індійський чай	5	2	60
Грузинський чай	2	6	87
Краснодарський чай	3	2	43
Прибуток від реалізації 1 т продукції, грн	32	29	

Потрібно скласти план виробництва чаю сортів А і Б з метою максимізації сумарного прибутку.

Задача 1.19. Для виготовлення столів і шаф використовується два види деревини: бук і вільха. Для виготовлення однієї шафи використовується 0,2 $м^3$ бука та 0,1 $м^3$ вільхи. Для виготовлення одного столу використовується 0,15 $м^3$ бука та 0,2 $м^3$ вільхи. Дохід майстерні від виробництва одного столу складає 12 ум. од., від виробництва однієї шафи – 15 ум. од. Визначте, скільки столів і шаф повинна виготовити майстерня, щоб забезпечити найбільшу рентабельність їх виробництва, якщо в розпорядженні майстерні є 60 $м^3$ бука та 40 $м^3$ вільхи.

Задача 1.20. Підприємство електронної промисловості випускає дві моделі радіоприймачів, причому кожна модель виробляється на окремій технологічній лінії. Добовий обсяг першої лінії становить 60 виробів, другої – 80. На радіоприймач першої моделі витрачається 15 однотипних елементів електронних схем, на радіоприймач другої моделі – 10 таких же елементів. Максимальний добовий запас використовуваних елементів дорівнює 950 одиниць. Прибутки від реалізації одного радіоприймача першої і другої моделей – 40 ум. од. і 20 ум. од., відповідно. Визначте оптимальні добові обсяги виробництва першої і другої моделей.

Задача 1.21. Підприємство випускає два види продукції: електричні печі (ЕП) і морозильні камери (МК). На виготовлення однієї ЕП потрібно затратити 4 кг пластика та 3 кг заліза. На виготовлення однієї МК потрібно затратити 5 кг пластика та 4 кг заліза. Виробництво забезпечене сировиною кожного типу в кількості 420 і 516 кг, відповідно. Ринкова ціна однієї ЕП становить 2 396 ум. од., а МК − 4 999 тис. ум. од. Складіть план виробництва виробів, що забезпечує максимальну виручку від їх реалізації.

Лабораторна робота 2
Задача лінійного програмування та методи її розв'язування: симплексний метод розв'язування ЗЛП

Мета роботи: набути практичних навичок для розв'язання задач лінійного програмування симплексним методом за допомогою стандартних процедур середовища MATLAB.

Основні задачі лабораторної роботи

1. Для заданої математичної моделі вихідної задачі запрограмувати у середовищі MATLAB алгоритм розв'язку останньої.

2. За допомогою вбудованою функції `optimset` зробити відповідні налаштування функції `linprog` середовища MATLAB таким чином, щоб знайти оптимальний розв'язок задачі за допомогою симплексного методу.

3. За допомогою вхідного параметра `options` установити додаткові налаштування функції `linprog`, вибрати різні числа допустимих ітерацій симплекс-методу, провести порівняльний аналіз отриманих результатів.

4. Для тривимірної графічної ілюстрації задачі зобразити поверхню, що оптимізується, та побудувати площини, що відповідають граничним умовам задачі.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

2.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

OPTIMSET — **Створити або відредагувати структуру параметрів опцій оптимізації**

Синтаксис

```
options = optimset('param1',value1,'param2',value2,...)
optimset
options = optimset
options = optimset(optimfun)
options = optimset(oldopts,'param1',value1,...)
options = optimset(oldopts,newopts)
```

Опис

`options = optimset ('param1', value1, 'param2', value2, ...)` створює структуру параметрів опцій оптимізації за замовчуванням в `options`, в який специфіковані параметри (`param`) мають специфіковані значення. Будь-який неспецифікований параметр установлюється як `[]` (параметри зі значенням `[]` вказують, що, коли опція передається в оптимізаційну функцію, для даних параметрів використовуються прийняті за замовчуванням значення). Для того щоб однозначно визначити ім'я параметра, досить тільки набрати початкові символи, які визначають відповідний параметр (така операція не допускається для імен параметрів):

`optimset` без вхідних і вихідних параметрів відображає повний список параметрів з їх дозволеними значеннями;

`options = optimset` (без вхідних аргументів) створює опції з опціонних структур, де всі поля встановлюються як `[]`;

`options = optimset (optimfun)` створює опції з опціонних структур з усіма іменами параметрів і прийнятими за замовчуванням значеннями, що належать до оптимізаційної функції `optimfun`;

`options = optimset (oldopts, 'param1', value1, ...)` створює копію для `oldopts`, модифікуючи специфіковані параметри за допомогою специфіковані значень;

`options = optimset (oldopts, newopts)` поєднує існуючу опціонну структуру `oldopts` з новою опціонною структурою `newopts`. Будь-який параметр у `newopts` з непорожніми значеннями перезаписується на відповідні старі параметри в `oldopts`.

Параметри

Докладна інформація про окремі параметри розміщена в табл. А.1 додатка А. У табл. 2.1 значення *у* {} указують на прийняті за замовчуванням значення. Водночас частина параметрів має різні прийняті за замовчуванням значення для різних функцій оптимізації і тому в деяких {} ніяких значень не показано.

Також можна переглянути параметри оптимізації і прийняті за замовчуванням значення, якщо набрати `optimset` у командній лінії.

Таблиця 2.1

Параметри оптимізації, що використовуються в крупно та середньомасштабному алгоритмах

Назва параметру	Значення параметру
Diagnostics	'on' | {'off'}
Display	'off' | 'iter' | 'final' | 'notify'
GradObj	'on' | {'off'}
Jacobian	'on' | {'off'}
LargeScale	'on' | {'off'}
MaxFunEvals	Додатне ціле
MaxIter	Додатне ціле
Simplex	'on' | {'off'}
TolFun	Додатний скаляр
TolX	Додатний скаляр

Згідно з табл. 2.1 для того, щоб встановити використання алгоритму симплексного методу, необхідно встановити опціонні параметри типу `'LargeScale'` в `'off'` та `'Simplex'` в `'on'`:

```
options = optimset('LargeScale', 'off', 'Simplex', 'on')
```

Далі викликається функція `linprog` з відповідними вхідними опціями й аргументами. Табл. 2.2 і 2.3 пояснюють ще деякі важливі параметри та їх значення функції `optimset`.

Таблиця 2.2

Параметри оптимізації, що використовуються в крупно та середньомасштабному алгоритмах

Назва параметру	Значення параметру
Hessian	`'on' | {'off'}`
HessMult	`function | {[]}`
HessPattern	Розріджена матриця | {Розріджена матриця з будь-яких}
JacobMult	`function | {[]}`
JacobPattern	Розріджена матриця | {Розріджена матриця з будь-яких}
MaxPCGIter	Додатне ціле | {Більше 1 і не перевершує `(n / 2)`}, де `n` є число елементів `x0`, тобто для стартової точки
PrecondBandWidth	Додатне ціле | `{0} | Inf`
TolPCG	Додатний скаляр | `{0.1}`
TypicalX	Вектор від будь-яких значень

Таблиця 2.3

Параметри оптимізації, що використовуються тільки в середньомасштабному алгоритмі

Назва параметру	Значення параметру
DerivativeCheck	`'on' | {'off'}`
DiffMaxChange	Додатний скаляр | `{1e-1}`
DiffMinChange	Додатний скаляр | `{1e-8}`
GoalsExactAchieve	Додатне скалярне ціле | `{0}`
GradConstr	`'on' | {'off'}`
HessUpdate	`{'bfgs'} | 'dfp' | 'gillmurray' | 'steepdesc'`
LevenbergMarquardt	`'on' | {'off'}`
LineSearchType	`'cubicpoly' | {'quadcubic'}`
MeritFunction	`'singleobj' | {'multiobj'}`
MinAbsMax	Додатне скалярне ціле | `{0}`

Нижче наведені деякі приклади використання функції `optimset`.

Перша команда викликається як `options` і створює структуру опцій оптимізації, де параметр відображення `Display` встановлюється як `'iter'` і параметр `TolFun` приймає значення `1e-8`:

```
options = optimset ( 'Display', 'iter', 'TolFun', 1e-8)
```

Друга команда викликається як `options` і створює копію структури опцій, змінюючи значення параметра `TolX` і зберігаючи нове значення в `optnew`:

```
optnew = optimset (options, 'TolX', 1e-4)
```

Третя команда повертає опції структури опцій оптимізації, які включають в себе імена параметрів і прийняті за умовчанням відповідні функції `linprog` значення:

```
options = optimset ('linprog')
```

2.2. Розв'язування типових задач у середовищі MATLAB

Приклад 2.1. Розв'язати таку задачу лінійного програмування симплексним методом:

$$z = -5x_1 + x_2 + x_3 \to \min;$$
$$\begin{cases} x_1 + x_2 \geq 4 \\ 5x_1 + x_2 + x_3 = 14; \end{cases}$$
$$x_j \geq 0,\ j = \overline{1,3}.$$

Розв'язання. Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `linprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clc % Очистити командне вікно
clear all % Видалення змінних і функцій з пам'яті
% Вхідні аргументи:
f = [-5; 1; 1];% Вектор коефіцієнтів функції цілі
%Матриця A і вектор b є, відповідно, коефіцієнтами матриці
%обмежень у вигляді лінійних нерівностей і вектора з правої
%частини:
A = [-1 -1 0];
b = [-4];
```

```
%Матриця Aeq і вектор beq є, відповідно, коефіцієнтами матриці
%обмежень у вигляді лінійних рівностей і вектора з правої частини:
Aeq = [5 1 1];
beq = [14];
lb = zeros(3, 1);%Нижній обмежувальний вектор
%За замовчуванням використовується алгоритм внутрішньої точки.
%Щоб вибрати симплекс-метод, потрібно написати:
options = optimset('LargeScale','off','Simplex','on');
%Виклик функції linprog:
[x,fval,exitflag,output,lambda] =
linprog(f,A,b,Aeq,beq,lb,[],[],options);
%Вихідні дані:
x
fval
exitflag
iter = output.iterations
algorithm = output.algorithm
ineqlin = lambda.ineqlin
eqlin = lambda.eqlin
low = lambda.lower
%========= Для графічної ілюстрації побудуємо поверхні
============
[X,Y] = meshgrid(0:0.05:3,0:0.05:2);
Z = 5.*X -Y;
mesh(X,Y,Z)
hold on
Z = 14 - 5.*X - Y;
mesh(X,Y,Z)
[X,Z] = meshgrid(0:0.05:3,-12:0.1:12);
Y = 4 - X;
mesh(X,Y,Z)
```

Результати роботи програми :

```
Optimization terminated.
x =
    2.5000
    1.5000
         0
fval = -11
exitflag = 1
iter = 0
algorithm = medium scale: simplex
ineqlin = 2.5000
eqlin = 1.5000
```

```
low =     0
          0
     2.5000
```

Отриманий опорний план $X^* = (2.5, 1.5, 0)$ є оптимальним, а значення цільової функції $Z_{\min} = -11$ є мінімальним. Як і в прикладі 1.1 з лабораторної роботи 1, середовище MATLAB дозволяє вивести повну інформацію про ітераційний процес розв'язання задачі. Так, у програмі був використаний алгоритм середньомасштабної оптимізації (`algorithm = medium scale: simplex`), який успішно завершив роботу (`exitflag = 1`). Оскільки `linprog` має активний набір методів і, таким чином, є варіантом добре відомого симплексного методу для лінійного програмування, то він знаходить початкове дозволене рішення шляхом первісного розв'язання задачі лінійного програмування. Ненульові елементи векторів у полях `lambda` вказують на активні обмеження у розв'язанні. У нашому випадку обидва обмеження у вигляді нерівності (в `lambda.ineqlin`) і рівності (в `lambda.eqlin`) і обидва обмеження за знаком (у `lambda.lower`) є активними обмеженнями.

Побудова площин, відповідних граничним умовам (рис. 2.1), виконується за допомогою введення команд у m-файлі після коментаря:

```
"%= Для графічної ілюстрації побудуємо поверхні ="
```

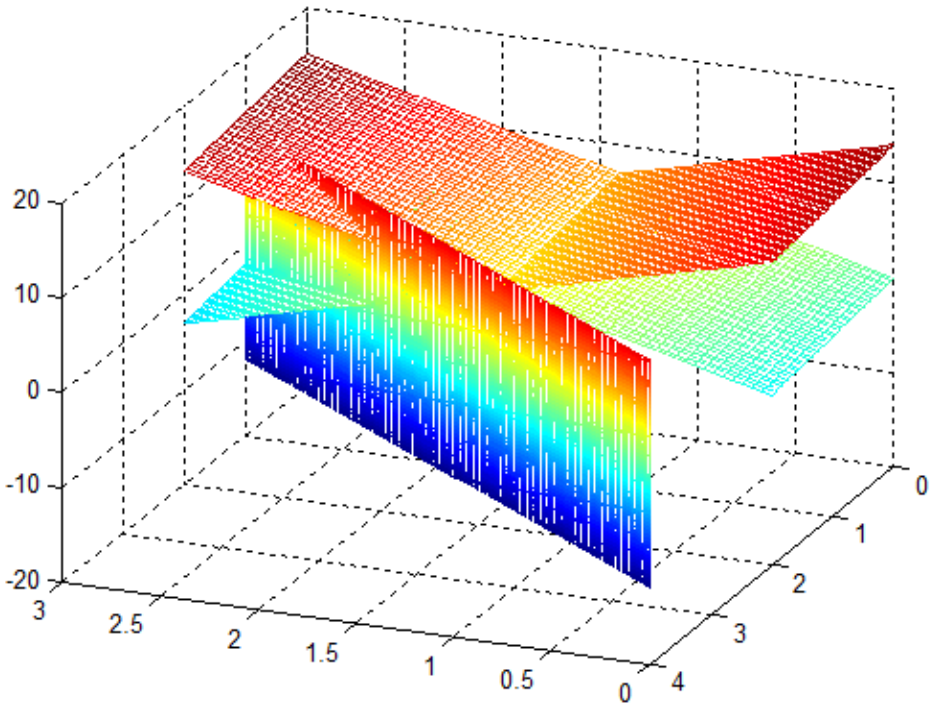

Рис. 2.1. **Перетин поверхні з площинами, що відповідають граничним умовам**

Для симплекс-методу допустима кількість ітерацій (MaxIter) за замовчуванням в 10 разів більше кількості змінних. Значення MaxIter можна змінити. Щоб встановити допустиму кількість ітерацій дорівненою, наприклад, 10, потрібно написати:

```
options = optimset ('LargeScale', 'off', 'Simplex', 'on', 'MaxIter', 10);
[X, fval] = linprog (f, A, b, Aeq, beq, lb, ub, [], options).
```

Якщо після виконання десятої ітерації рішення не буде знайдено, параметр exitflag стане нульовим, і на екрані з'явиться повідомлення:

```
Maximum number of iterations exceeded;
increase options.MaxIter.
```

Запитання для самоперевірки

1. Як у MATLAB називають функцію, що реалізує симплекс-метод?
2. Опишіть процес створення і редагування структури параметрів опцій оптимізації.
3. Яке число допустимих ітерацій за замовчуванням встановлено в симплекс-методі функції linprog середовища MATLAB?
4. Наведіть алгоритм симплексного методу розв'язання задач ЛП.
5. Назвіть вхідні та вихідні параметри функції optimset.
6. Як перетворювати обмеження-нерівності вихідної задачі лінійного програмування, що має вигляд "≥", якщо функція linprog працює лише з обмеженнями виду "≤"?
7. Які функції MATLAB використовують для зображення тривимірного розв'язання задачі ЛП?

Завдання до лабораторної роботи

1. Для заданих математичних моделей вихідних задач запрограмувати у середовищі MATLAB алгоритм розв'язку останніх.
2. За допомогою вбудованою функції optimset зробити відповідні налаштування функції linprog середовища MATLAB таким чином, щоб знайти оптимальний розв'язок задачі за допомогою симплексного методу.

3. За допомогою вхідного параметра `options` установити додаткові налаштування функції `linprog`, вибрати різні числа допустимих ітерацій симплекс-методу, провести порівняльний аналіз отриманих результатів.

4. Для тривимірної графічної ілюстрації задачі зобразити поверхню, що оптимізується, та побудувати площини, що відповідають граничним умовам задачі.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Варіанти завдань наведені у табл. 2.4.

Таблиця 2.4

Варіанти завдань до лабораторної роботи 2

№ варіанта та математична модель ЗЛП	№ варіанта та математична модель ЗЛП	№ варіанта та математична модель ЗЛП
1	2	3
$z = x_1 + 4x_2 + x_3 \to \max$ 1. $\begin{cases} -x_1 + 2x_2 + x_3 = 4 \\ 3x_1 + x_2 + 2x_3 \le 9 \\ 2x_1 + 3x_2 + x_3 \ge 6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 2x_1 + x_2 - x_3 \to \min$ 2. $\begin{cases} 2x_1 + x_2 - x_3 \ge 5 \\ x_1 + 2x_2 + x_3 \le 7 \\ x_1 - x_2 + 2x_3 = 1 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 - x_2 + x_3 \to \max$ 3. $\begin{cases} 4x_1 + 2x_2 + x_3 \ge 6 \\ -x_1 + x_2 + x_3 = 1 \\ x_1 - x_2 + 4x_3 \le 24 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = 5x_1 + x_2 + x_3 \to \max$ 4. $\begin{cases} x_1 + x_2 + x_3 \ge 3 \\ x_1 + 2x_2 + 2x_3 = 4 \\ 3x_1 + 4x_2 + 2x_3 \le 12 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 - x_3 \to \max$ 5. $\begin{cases} 3x_1 + x_2 + 2x_3 \ge 6 \\ x_1 + x_2 + x_3 = 4 \\ x_1 - 3x_2 + x_3 \le -4 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 - x_2 - x_3 \to \max$ 6. $\begin{cases} 3x_1 + x_2 + x_3 \ge 6 \\ x_1 + 3x_2 + x_3 = 10 \\ x_1 - 3x_2 + x_3 \le -2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 + x_2 + 3x_3 \to \max$ 7. $\begin{cases} x_1 - 3x_2 + 2x_3 = 3 \\ 2x_1 + 4x_2 + x_3 \le 18 \\ -x_1 + x_2 + x_3 \ge 10 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 4x_1 - 3x_2 - x_3 \to \max$ 8. $\begin{cases} 4x_1 + x_2 + x_3 \ge 8 \\ 2x_1 + x_2 - x_3 = 6 \\ x_1 - 3x_2 - x_3 \ge -4 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 + x_3 \to \max$ 9. $\begin{cases} 4x_1 - x_2 - 2x_3 = 3 \\ x_1 + 3x_2 + x_3 \ge 4 \\ 3x_1 - x_2 + x_3 \le 12 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 - 3x_2 - 2x_3 \to \max$ 10. $\begin{cases} 3x_1 + x_2 - 2x_3 \ge 13 \\ x_1 - 3x_2 + x_3 = 1 \\ x_1 + 2x_2 + 3x_3 \le 11 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 3x_1 + 2x_2 + 2x_3 \to \min$ 11. $\begin{cases} 3x_1 + 2x_2 - 2x_3 \ge 4 \\ x_1 - 6x_2 - 3x_3 = 2 \\ -x_1 + x_2 - x_3 \ge 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 + x_3 \to \max$ 12. $\begin{cases} -x_1 + x_2 + 2x_3 \le 2 \\ -x_1 + 2x_2 + x_3 \le 4 \\ x_1 + 2x_3 = 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$

Закінчення табл. 2.4

1	2	3
$z = x_1 + x_2 + x_3 \to \max$ 13. $\begin{cases} -x_1 - x_2 + x_3 \le -1 \\ x_1 + x_2 + x_3 = 3 \\ x_1 + x_3 \le 1 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 2x_1 + x_2 + x_3 \to \max$ 14. $\begin{cases} x_1 + 2x_2 - x_3 \ge 2 \\ -2x_1 + x_2 + 2x_3 = 2 \\ -2x_1 - x_2 + x_3 \ge -6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 - x_2 + x_3 \to \max$ 15. $\begin{cases} x_1 + 2x_2 + x_3 \ge 5 \\ x_1 + x_2 + x_3 \ge 2 \\ x_1 + x_2 + 3x_3 \ge 4 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = -x_1 - x_2 + x_3 \to \min$ 16. $\begin{cases} x_1 + x_2 + x_3 \ge 3 \\ x_1 + x_3 \le 2 \\ x_1 - x_2 - x_3 = -1 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 - x_3 \to \min$ 17. $\begin{cases} 3x_1 + x_2 + 2x_3 \ge 12 \\ x_1 + x_2 + x_3 = 8 \\ 2x_1 - 3x_2 + x_3 \ge 8 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 + x_3 \to \min$ 18. $\begin{cases} x_1 + 2x_2 + 2x_3 \ge 5 \\ x_1 + x_2 + x_3 = 2 \\ x_1 + x_2 + x_3 \ge -2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = 5x_1 + x_2 + x_3 \to \max$ 19. $\begin{cases} x_1 + x_2 + x_3 = 3 \\ 2x_1 + x_2 + x_3 \ge 4 \\ 3x_1 + 2x_2 - 2x_3 \le 12 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 - x_2 - x_3 \to \max$ 20. $\begin{cases} 2x_1 + 2x_2 + x_3 \ge 6 \\ x_1 + x_2 + 2x_3 = 1 \\ -x_1 + 3x_2 + x_3 \ge -7 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 - x_2 - x_3 \to \max$ 21. $\begin{cases} 3x_1 + x_2 + x_3 \ge 6 \\ x_1 + x_2 + x_3 = 4 \\ 3x_1 - 3x_2 + x_3 \ge -4 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 + x_2 - 4x_3 \to \min$ 22. $\begin{cases} x_1 + x_2 - x_3 = 9 \\ 4x_1 + 7x_2 + x_3 \ge 11 \\ 3x_1 - x_2 - 2x_3 \le 12 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 + x_3 \to \max$ 23. $\begin{cases} x_1 + 4x_2 + x_3 = 3 \\ 2x_1 + 3x_2 + x_3 \ge 4 \\ x_1 - 5x_2 - 2x_3 \le 18 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + 2x_2 + 3x_3 \to \min$ 24. $\begin{cases} x_1 + 4x_2 + x_3 \le 3 \\ 2x_1 + 6x_2 + 7x_3 \ge 4 \\ -x_1 + 2x_2 - 2x_3 \le 1 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$

Лабораторна робота 3
Теорія двоїстості й аналіз лінійних моделей економічних оптимізаційних задач

Мета роботи: набути практичних навичок щодо побудови математичної моделі двоїстої задачі та визначення оптимального плану вихідної задачі за розв'язком двоїстої у програмному середовищі MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель вихідної задачі. Для заданої математичної моделі вихідної задачі запрограмувати у середовищі MATLAB алгоритм розв'язку останньої.

2. Для заданої задачі лінійного програмування побудувати двоїсту задачу та, використовуючи основні теореми двоїстості, записати оптимальний розв'язок для обох.

3. Використовуючи функцію `optimget`, отримати значення параметрів опцій оптимізації. Перевірити, що отриманий в пункті 2 оптимальний розв'язок, є оптимальним планом двоїстої задачі, який задовольняє теоремам і співвідношенням двоїстості.

4. Привести економічну інтерпретацію двоїстих задач і двоїстих оцінок, використовуючи основні теореми двоїстості.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

3.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

OPTIMGET — **Отримати значення параметрів опцій оптимізації**

Синтаксис

```
val = optimget(options,'param')

val = optimget(options,'param',default)
```

Опис

`val = optimget (options, 'param')` повертає значення спеціалізованого параметра в структурі опцій оптимізації. Для того щоб однозначно визначити ім'я параметра, необхідно лише надрукувати достатню кількість визначальних символів. Цей випадок ігнорується для параметричних імен.

`val = optimget (options, 'param', default)` за замовчуванням повертає структуру опцій, якщо в оптимізаційних опціях не визначені спеціалізовані параметри. Зазначимо, що ця функціональна форма в першу чергу використовується для оптимізаційних функцій.

Приклади використання

Функція `optimget` надає інформацію про вихідний параметр `lambda` функції `linprog`, в якому міститься розв'язання двоїстої задачі лінійного програмування. Параметр `lambda` складається з чотирьох масивів: `lambda.ineqlin`, `lambda.eqlin`, `lambda.upper`, `lambda.lower`. У цих масивах знаходяться змінні двоїстої задачі, приписані обмеженням-нерівностям, обмеженням-рівностям, обмеженням на план зверху та знизу, відповідно.

Розглянемо задачу лінійного програмування в загальному вигляді:

$$f = c_N \times x_N \to inf;$$
$$D_{M_1,N} \times x_N \geq B_{M_1};$$
$$D_{M_2,N} \times x_N = B_{M_2};$$
$$-x \geq v_N;$$
$$x_N \geq w_N$$

де $N = \overline{1,n}, M_1 = \overline{1,s}, M_2 = s + \overline{1,t}$.

Аналогічно з лабораторною роботою 1 подамо нашу задачу відповідно до заданої форми роботи команди.

$f = c_N \times x_N \to \inf$	Вектор коефіцієнтів цільової функції: `f=c[N];`

$D_{M_1,N} \times x_N \geq B_{M_1}$	Матриця коефіцієнтів системи лінійних нерівностей:
	`A=-D[M1, N];`
	Вектор вільних членів системи лінійних нерівностей:
	`b=-B[M1];`

$D_{M_2,N} \times x_N = B_{M_2}$	Матриця коефіцієнтів системи лінійних рівностей:
	`Aeq=-D[M2, N];`
	Вектор вільних членів системи лінійних рівностей:
	`Beq=-B[M2];`

	Вектор нижніх меж для змінних x_1, x_2:
$x_N \geq w_N$	`lb=w[N];`
	Вектор верхніх меж для змінних x_1, x_2:
$-x \geq v_N$	`ub=v[N];`

Спочатку розглянемо випадок, коли `w[N]`=∅ або `w[N]` не заданий. Змінних у двоїстій задачі буде `t + n`. Змінні двоїстої задачі містяться в параметрі `lambda`. Їх можна знайти за формулами:

```
ui = lambda.ineqlin(i), i = 1, ..., s,
ui = lambda.eqlin(i – s), i = s + 1, ..., t,
ui = lambda.upper(i – t), i = t + 1, ..., t + n.
```

Зауважимо, що:

1) якщо `M1` = ∅, то вважаємо `s = 0` і не використовуємо перше співвідношення (масив `lambda.ineqlin` порожній);

2) якщо `M2` = ∅, то вважаємо `t = s` і не використовуємо друге співвідношення (масив `lambda.eqlin` порожній);

3) якщо не заданий вектор верхніх меж `v[N]`, то третє співвідношення не використовується (масив `lambda.upper` нульовий).

Тепер припустимо, що `w[N]` ≠ ∅. Нехай $w[k_i] \neq 0$, $i = 1,\ldots,l$, $l \leq n$. Тим самим виділяються індекси знакових обмежень $N \setminus \{k_1, \ldots, k_l\}$. Змінних у двоїстій задачі буде `t + n + l`. Їх можна знайти за формулами:

```
ui = lambda.ineqlin(i), i = 1, ..., s,
ui = lambda.eqlin(i – s), i = s + 1, ..., t,
ui = lambda.upper(i – t), i = t + 1, ..., t + n,
ui = lambda.lower(k_{i-t-n}), i = t + n + 1, ..., t + n + l.
```

Якщо не заданий вектор верхніх меж `v[N]`, то третє співвідношення не використовується (масив `lambda.upper` нульовий), а четверте набуває вигляду:
`ui = lambda.lower(k_{i-t}), i = t + 1, ..., t + l.`

Приклад 3.1. Розглянемо задачу лінійного програмування:

$$f(x) = x_1 + x_2 \to \inf;$$
$$\begin{cases} x_1 - x_2 \geq 2 \\ x_1 - 2x_2 \geq 1 \\ -x_1 \geq -4 \\ -x_2 \geq -4; \end{cases}$$
$$x_1 \geq 0,\ x_2 \geq 0.$$

Розв'язання. Складемо двоїсту задачу:

$$f^*(y) = 2y_1 + y_2 - 4y_3 - 4y_4 \to \sup;$$
$$\begin{cases} y_1 + y_2 - y_3 \leq 1 \\ -y_1 - 2y_2 - y_4 \leq 1; \end{cases}$$
$$y_j \geq 0,\ j = \overline{1,4}.$$

Розв'яжемо вихідну задачу в середовищі MATLAB. Програма буде виглядати таким чином:

```
clear all
close all
clc
C = [1 1];
D = [1 -1; 1 -2];
B = [2 1];
lb = zeros(2,1);
ub = [4 4];
f = C;
A = -D;
b = -B;
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output,lambda]=linprog(f,A,b,[],[],lb,ub,[],optio
ns);
x
fval
lambda.ineqlin
lambda.eqlin
lambda.upper
lambda.lower
```

На виході отримаємо оптимальний план $X^* = (2, 0)$, мінімальне значення цільової функції $f_{\min} = 2$ і параметр `lambda`:

```
Optimization terminated.

x =  2
     0
fval = 2
lambda.ineqlin =
     1
     0
lambda.eqlin = Empty matrix: 0-by-1

lambda.upper = 0
               0

lambda.lower = 0
               2
```

Зауважимо, що у вихідній задачі прикладу 3.1 `t = s` і `w[N] =` $\varnothing$. За формулами матимемо: $y_1 = y_2 = y_3 = y_4 = 0$. Легко перевірити, що це план двоїстої задачі, що задовольняє умовам та співвідношенням двоїстості $f_{\min}(x) = 2 = f^*(y)$.

Приклад 3.2. Розглянемо задачу лінійного програмування:

$$f(x) = x_1 + x_2 \to \inf;$$
$$\begin{cases} x_1 - x_2 \geq 2 \\ x_1 - 2x_2 \geq 1 \\ x_1 \geq 0 \\ x_2 \geq -1. \end{cases}$$

Розв'язання. Складемо двоїсту задачу:

$$f^*(y) = 2y_1 + y_2 - y_3 \to \sup;$$
$$\begin{cases} y_1 + y_2 + y_3 \leq 1 \\ -y_1 - 2y_2 - y_3 = 1; \end{cases}$$
$$y_j \geq 0,\ j = \overline{1,3}.$$

Розв'яжемо вихідну задачу в середовищі MATLAB. Програма буде виглядати таким чином:

```
clear all
close all
clc
C = [1 1];
D = [1 -1; 1 -2];
B = [2 1];
lb = [0 -1];
f = C;
A = -D;
b = -B;
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output,lambda]=linprog(f,A,b,[],[],lb,[],[],optio
ns);
x
fval
lambda.ineqlin
lambda.eqlin
lambda.upper
lambda.lower
```

На виході отримаємо оптимальний план $X^* = (1, -1)$, мінімальне значення цільової функції $f_{\min} = 0$ та параметр `lambda`:

```
Optimization terminated.

x =  1
    -1
fval = 0
lambda.ineqlin =
     1
     0
lambda.eqlin = Empty matrix: 0-by-1

lambda.upper = 0
               0

lambda.lower = 0
               2
```

Зауважимо, що у вихідній задачі прикладу 3.2. відсутній `v[N]` = ∅ і `w[1]` = 0, `w[2]` ≠ 0 (тобто `l` = 1, k_1 = 2). За формулами матимемо: $y_1 = 1, y_2 = 0, y_3 = 2$. Легко перевірити, що це план двоїстої задачі, що задовольняє умовам та співвідношенням двоїстості $f_{\min}(x) = 0 = f^*(y)$.

3.2. Розв'язування типових задач у середовищі MATLAB

Приклад 3.3. Для виробництва трьох видів виробів *A*, *B* і *C* використовується три різних види сировини. Кожен з них може бути використаний в кількості, відповідно, не більшій 180, 210 і 244 кг. Норми витрат кожного з видів сировини на одиницю продукції та ціна одиниці продукції кожного виду наведені в табл. 3.1.

Таблиця 3.1

Вихідні дані

Види сировини	Норми витрат сировини (кг) на одиницю продукції			Запаси сировини
	A	*B*	*C*	
I	4	2	1	180
II	3	1	3	210
III	1	2	5	244
Ціна одиниці продукції (ум. од.)	10	14	12	

Визначте план випуску продукції, за якого забезпечується її максимальна вартість. Дайте оцінку кожному з видів сировини, що використовуються для виробництва продукції. Оцінки, приписувані кожному з видів сировини, повинні бути такими, щоб загальна оцінка всієї використовуваної сировини була мінімальною, а сумарна оцінка сировини, використовуваної на виробництво одиниці продукції кожного виду, − не менше ціни одиниці продукції даного виду.

Розв'язання. Припустимо, що виробляється x_1 виробів *A*, x_2 виробів *B* і x_3 виробів *C*. Для визначення оптимального плану виробництва потрібно розв'язати задачу, яка полягає в максимізації цільової функції

$$f = 10x_1 + 14x_2 + 12x_3 \to \max$$

за таких умов:

$$\begin{cases} 4x_1 + 2x_2 + x_3 \le 180 \\ 3x_1 + x_2 + 3x_3 \le 210 \\ x_1 + 2x_2 + 5x_3 \le 244; \end{cases}$$
$$x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0.$$

Припишемо кожному з видів сировини, що використовуються для виробництва продукції, подвійну оцінку, відповідно рівну y_1, y_2 та y_3. Тоді загальна оцінка сировини, використовуваної на виробництво продукції, складе

$$f^* = 180y_1 + 210y_2 + 244y_3 \to \min.$$

Згідно з умовою, двоїсті оцінки повинні бути такими, щоб загальна оцінка сировини, використовуваної на виробництво одиниці продукції кожного виду, була не меншою ціни одиниці продукції даного виду, тобто y_1, y_2 та y_3 повинні задовольняти такій системі нерівностей:

$$\begin{cases} 4y_1 + 3y_2 + y_3 \ge 10 \\ 2y_1 + y_2 + 2y_3 \ge 14 \\ y_1 + 3y_2 + 5y_3 \ge 12; \end{cases}$$
$$y_1 \ge 0,\ y_2 \ge 0,\ y_3 \ge 0.$$

Як видно, задачі утворюють симетричну пару двоїстих задач. Розв'язок прямої задачі дає оптимальний план виробництва виробів *A*, *B* і *C*, а розв'язок двоїстої − оптимальну систему оцінок сировини, використовуваної для виробництва цих виробів. Щоб знайти розв'язок цих задач, слід спочатку знайти розв'язок однієї з них. Оскільки система обмежень прямої задачі містить лише нерівності виду "≤", то краще спочатку знайти розв'язок цієї задачі. Її розв'язок наведено у відповідному m-файлі, який містить такий програмний код:

```
clear all
close all
clc
A = [4 2 1; 3 1 3; 1 2 5];
b = [180 210 244];
lb = zeros(3,1);
f = [-10 -14 -12];
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output,lambda]=linprog(f,A,b,[],[],lb,[],[],options);
```

```
x = x
f=-fval
y1 = lambda.ineqlin(1)
y2 = lambda.ineqlin(2)
y3 = lambda.ineqlin(3)
```

Результати роботи програми:

```
Optimization terminated.
x =
     0
    82
    16

f = 1340

y1 = 5.7500
y2 = 0
y3 = 1.2500
```

З роботи програми видно, що оптимальним планом виробництва виробів є такий, за якого виготовляється 82 вироби *B* і 16 виробів *C*. За цим планом виробництва залишається невикористаним 80 кг сировини II виду, а загальна вартість виробів дорівнює 1 340 ум. од. З роботи програми також видно, що оптимальним розв'язком двоїстої задачі є $y_1 = 5{,}75,\ y_2 = 0,\ y_3 = 1{,}25$.

Змінні y_1 та y_3 позначають умовні двоїсті оцінки одиниці сировини, відповідно, I і III видів. Ці оцінки відмінні від нуля, а сировина I і III видів повністю використовується за оптимального плану виробництва продукції. Подвійна оцінка одиниці сировини II виду дорівнює нулю. Цей вид сировини не повністю використовується за оптимального плану виробництва продукції.

Таким чином, додатну двоїсту оцінку мають лише ті види сировини, які повністю використовуються за оптимальним планом виробництва виробів. Тому двоїсті оцінки визначають дефіцитність використовуваної підприємством сировини. Більше того, величина двоїстої оцінки показує, на скільки зростає максимальне значення цільової функції прямої задачі зі збільшенням кількості сировини відповідного виду на 1 кг. Так, збільшення кількості сировини I виду на 1 кг призведе до того, що з'явиться можливість знайти новий оптимальний план виробництва виробів, за якого загальна вартість виготовленої продукції зросте на 5,75 ум. од. і дорівнюватиме 1 340 + 5,75 = 1 345,75 ум. од. Точно так

же збільшення на 1 кг сировини III виду дозволить знайти новий оптимальний план виробництва виробів, коли загальна вартість виготовленої продукції зросте на 1,25 ум од. і складе 1 340 + 1,25 = 1 341,25 ум од.

Продовжимо розгляд оптимальних двоїстих оцінок. Обчислюючи мінімальне значення цільової функції двоїстої задачі та використовуючи наступний програмний код:

```
F=b*[y1; y2; y3],
```

у результаті отримаємо $f^* = 1340$. Бачимо, що це значення збігається з максимальним значенням цільової функції вихідної задачі. Підстановкою оптимальних двоїстих оцінок у систему обмежень двоїстої задачі отримуємо:

$$\begin{cases} 4 \cdot 5{,}75 + 1{,}25 > 10 \\ 2 \cdot 5{,}75 + 2 \cdot 1{,}25 = 14 \\ 5{,}75 + 5 \cdot 1{,}25 = 12. \end{cases}$$

Перше обмеження двоїстої задачі виконується як чітка нерівність. Це означає, що двоїста оцінка сировини, використовуваної на виробництво одного виробу виду *A*, більша за ціну цього виробу; отже, випускати вироби виду *A* невигідно. Його виробництво не передбачено оптимальним планом прямої задачі. Друге та третє обмеження двоїстої задачі виконуються як строгі рівності. Це означає, що двоїсті оцінки сировини, використовуваної для виробництва одиниці, відповідно, виробів *B* і *C*, в точності дорівнюють їх цінами. Тому випускати ці два види продукції за двоїстими оцінками економічно доцільно. Їх виробництво передбачено оптимальним планом прямої задачі.

Таким чином, двоїсті оцінки тісно пов'язані з оптимальним планом прямої задачі. Будь-яка зміна вихідних даних прямої задачі може вплинути як на її оптимальний план, так і на систему оптимальних двоїстих оцінок.

Запитання для самоперевірки

1. Як у MATLAB називають функцію, що допомагає реалізувати двоїстий симплекс-метод?

2. Опишіть процес створення і редагування структури параметрів опцій оптимізації для програмної реалізації прямої та двоїстої задач.

3. Назвіть вхідні та вихідні параметри функції `optimget`.

4. Як отримати розв'язок двоїстої задачі лінійного програмування через розв'язання прямої задачі?

5. Опишіть процес отримання:

а) оптимального плану двоїстої задачі;

б) оцінок ресурсів (двоїстих оцінок);

в) дефіцитних і недефіцитних (надлишкових) ресурсів;

г) обґрунтування ефективності оптимального плану.

Завдання до лабораторної роботи

Для виробництва трьох видів виробів *A*, *B* і *C* використовують три різних види сировини. Кожен з видів може бути використаний в кількості, відповідно, не більшій S_1, S_2, S_3 кг. Норми витрат кожного з видів сировини на одиницю продукції даного виду та ціна одиниці продукції кожного виду наведені в табл. 3.2 для кожного варіанта. Визначте план випуску продукції, за якого забезпечується її максимальна вартість, і оцініть кожен з видів сировини, що використовується для виробництва продукції. Для цього у середовищі MATLAB виконайте такі дії:

1) побудуйте математичну модель вихідної задачі. Для заданої математичної моделі вихідної задачі запрограмуйте у середовищі MATLAB алгоритм розв'язку останньої;

2) для заданої задачі лінійного програмування побудуйте двоїсту задачу і, використовуючи основні теореми двоїстості, запишіть оптимальний розв'язок для обох;

3) використовуючи функцію `optimget`, отримайте значення параметрів опцій оптимізації. Перевірте, що отриманий в пункті 2 оптимальний розв'язок, є оптимальним планом двоїстої задачі, який задовольняє теоремам і співвідношенням двоїстості;

4) наведіть економічну інтерпретацію двоїстих задач і двоїстих оцінок, використовуючи основні теореми двоїстості (оцінки, приписувані кожному з видів сировини, повинні бути такими, щоб оцінка всієї використовуваної сировини була мінімальною, а сумарна оцінка сировини, використовуваної на виробництво одиниці продукції кожного виду, − не менше ціни одиниці продукції даного виду).

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Таблиця 3.2

Варіанти завдань до лабораторної роботи 2

Номер варіанта	Вид сировини	Норми витрат сировини (кг) на одиницю продукції			Запаси сировини, S_i
		A	*B*	*C*	
1	2	3	4	5	6
1	I	1	2	1	280
	II	2	1	3	110
	III	1	2	3	154
	Ціна одиниці продукції (ум. од.)	6	8	10	
2	I	5	3	1	380
	II	1	1	2	200
	III	1	1	3	184
	Ціна одиниці продукції (ум. од.)	8	10	7	
3	I	4	2	1	230
	II	2	6	3	190
	III	2	2	3	254
	Ціна одиниці продукції (ум. од.)	6	11	7	
4	I	1	2	1	130
	II	2	2	3	390
	III	2	2	1	100
	Ціна одиниці продукції (ум. од.)	8	5	7	
5	I	2	2	1	211
	II	1	5	4	271
	III	3	2	3	125
	Ціна одиниці продукції (ум. од.)	6	11	7	
6	I	1	2	3	230
	II	5	6	4	210
	III	2	1	1	160
	Ціна одиниці продукції (ум. од.)	5	4	6	
7	I	1	2	3	310
	II	3	1	2	113
	III	3	2	4	131
	Ціна одиниці продукції (ум. од.)	5	6	2	

Продовження табл. 3.2

1	2	3	4	5	6
8	I	3	1	1	223
	II	4	5	7	423
	III	1	1	1	100
	Ціна одиниці продукції (ум. од.)	1	4	2	
9	I	2	3	6	191
	II	1	2	3	123
	III	2	3	1	180
	Ціна одиниці продукції (ум. од.)	1	5	2	
10	I	1	1	1	420
	II	5	2	3	373
	III	2	1	4	375
	Ціна одиниці продукції (ум. од.)	1	4	6	
11	I	4	2	5	680
	II	6	7	8	350
	III	1	1	4	740
	Ціна одиниці продукції (ум. од.)	7	1	3	
12	I	1	2	1	381
	II	6	1	8	252
	III	1	5	4	443
	Ціна одиниці продукції (ум. од.)	8	6	4	
13	I	1	5	1	510
	II	7	5	8	480
	III	1	1	1	290
	Ціна одиниці продукції (ум. од.)	9	10	2	
14	I	5	3	3	110
	II	1	1	2	206
	III	4	1	4	330
	Ціна одиниці продукції (ум. од.)	9	5	7	
15	I	1	1	1	220
	II	3	5	4	420
	III	2	2	1	110
	Ціна одиниці продукції (ум. од.)	2	4	1	

Закінчення табл. 3.2

1	2	3	4	5	6
16	I	5	3	6	320
	II	7	2	3	680
	III	2	3	1	195
	Ціна одиниці продукції (ум. од.)	8	7	4	
17	I	1	1	9	420
	II	7	2	3	510
	III	2	5	4	290
	Ціна одиниці продукції (ум. од.)	6	7	10	
18	I	1	7	9	460
	II	2	6	8	410
	III	3	5	4	520
	Ціна одиниці продукції (ум. од.)	5	10	12	
19	I	1	2	1	383
	II	5	1	3	458
	III	1	5	4	291
	Ціна одиниці продукції (ум. од.)	5	5	4	
20	I	1	8	1	515
	II	4	5	3	485
	III	1	1	1	250
	Ціна одиниці продукції (ум. од.)	6	7	2	

Лабораторна робота 4
Транспортна задача

Мета роботи: набуття практичних навичок щодо: побудови математичних моделей транспортних задач (ТЗ) і задач, що належать до них; а також розв'язку у програмному середовищі MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель вихідних задач лабораторної роботи.

2. Для заданих математичних моделей вихідних задач запрограмувати у середовищі MATLAB алгоритм розв'язку останніх.

3. Провести економічний аналіз отриманого розв'язку.

4. Розглянути на прикладі задачі оптимального розподілу автомобілів на маршрутах узагальнення ТЗ, а саме – задачу про призначення та її практичну реалізацію у програмному середовищі MATLAB.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

4.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

Для розв'язання транспортної задачі у програмному середовищі MATLAB використовується функція `linprog`, опис якої детально наведено у лабораторній роботі 1. Слід детально розглянути варіанти програмного коду, які слід застосовувати для розв'язання того чи іншого типу ТЗ.

Приклад 4.1 (транспортна задача класична). Є m постачальників і n споживачів однорідної продукції. Відомі питомі витрати на доставку одиниці продукції від кожного постачальника кожному споживачеві. Запаси продукції у постачальників обмежені. Відомі також потреби в продукції кожного споживача. Причому потреби споживача дорівнюють запасам постачальників (збалансована задача). Необхідно визначити такий план перевезення продукції від постачальників до споживачів, за якого загальні витрати на перевезення будуть мінімальними.

Таким чином, потрібно знайти розв'язок транспортної задачі з вихідними даними, заданими в табл. 4.1.

Таблиця 4.1

Вихідні данні класичної транспортної задачі

Постачальники A_i	Споживачі B_j				Запаси
	1	2	3	4	
1	1	4	4	2	150
2	3	6	3	9	300
3	4	8	6	2	250
4	1	5	5	13	150
Потреби	150	200	100	400	

Розв'язання. Спочатку необхідно переконатися у виконанні рівняння балансу. Зокрема, для наведеного прикладу виконується рівність

$$\sum_{i=1}^{4} A_i = \sum_{j=1}^{4} B_j = 850.$$

Складемо математичну модель ТЗ і наведемо основну інтерпретацію моделі ТЗ згідно з синтаксисом функції `linprog` середовища MATLAB.

$$f = 1\cdot x_{11} + 4\cdot x_{12} + 4\cdot x_{13} + 2\cdot x_{14} + \\ + 3\cdot x_{21} + 6\cdot x_{22} + 3\cdot x_{23} + 9\cdot x_{24} + \\ + 4\cdot x_{31} + 8\cdot x_{32} + 6\cdot x_{33} + 2\cdot x_{34} + \\ + 1\cdot x_{41} + 5\cdot x_{42} + 5\cdot x_{43} + 13\cdot x_{44} \to \min$$

Від кожного постачальника можна вивести обсяг продукції не більше наявної кількості:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 150 \\ x_{21} + x_{22} + x_{23} + x_{24} = 300 \\ x_{31} + x_{32} + x_{33} + x_{34} = 250 \\ x_{41} + x_{42} + x_{43} + x_{44} = 150 \end{cases}$$

Потреба кожного споживача в продукції повинна бути задоволена:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 150 \\ x_{12} + x_{22} + x_{32} + x_{42} = 200 \\ x_{13} + x_{23} + x_{33} + x_{43} = 100 \\ x_{14} + x_{24} + x_{34} + x_{44} = 400 \end{cases}$$

$x_{ij} \ge 0, i = \overline{1,4},\ j = \overline{1,4}$

Вектор коефіцієнтів цільової функції:

```
f=[1; 4; 4; 2;
   3; 6; 3; 9;
   4; 8; 6; 2;
   1; 5; 5; 13];
```

Матриця коефіцієнтів системи лінійних нерівностей:

```
A=[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1


1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1];
```

Вектор вільних членів системи лінійних нерівностей:

```
b=[150; 300; 250; 150;
150; 200; 100; 400];
```

Вектор нижніх меж для змінних $x_{ij}, i = \overline{1,4},\ j = \overline{1,4}$:

```
lb = zeros(16, 1);
```

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `linprog`, наведені у відповідній програмі (m-файл), який виглядає так:

```
% Розв'язок транспортної задачі
% шляхом її приведення до задачі лінійного програмування
clc
clear all
% Вектор коефіцієнтів
f = [1; 4; 4; 2; 3; 6; 3; 9; 4; 8; 6; 2; 1; 5; 5; 13];
```

```
% Бінарні коефіцієнти в обмеженнях
Aeq = [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1];

% Праві частини обмежень
Beq = [150; 300; 250; 150; 150; 200; 100; 400];

% Нульовий вектор для завдання умови невід'ємності
% обсягів перевезень
lb = zeros (16, 1);
% Функція обчислення шуканих обсягів перевезень
[X fval] = linprog (f, [], [], Aeq, Beq, lb);
Xopt = reshape (x, 4,4)'
Fopt = fval
```

Результати роботи програми:

```
Optimization terminated.

Xopt =

    0.0000    0.0000    0.0000  150.0000
    0.0000  200.0000  100.0000    0.0000
    0.0000    0.0000    0.0000  250.0000
  150.0000    0.0000    0.0000    0.0000

Fopt = 2.4500e+003
```

Таким чином, мінімальні витрати складуть $f_{\min} = 2450$ ум. од. відповідно до оптимального плану:

$$X_{opt} = \begin{pmatrix} 0 & 0 & 0 & 150 \\ 0 & 200 & 100 & 0 \\ 0 & 0 & 0 & 250 \\ 150 & 0 & 0 & 0 \end{pmatrix}.$$

Приклад 4.2 (транспортна задача з фіктивним споживачем). Є *m* постачальників і *n* споживачів однорідної продукції. Відомі питомі витрати на доставку одиниці продукції від кожного постачальника кожному споживачеві. Запаси продукції у постачальників обмежені. Відомі також потреби в продукції кожного споживача. Але, на відміну від попереднього прикладу, запаси постачальників перевищують потреби споживачів. Необхідно визначити такий план перевезення продукції від постачальників до споживачів, за якого загальні витрати на перевезення будуть мінімальними.

Таким чином, потрібно знайти розв'язок транспортної задачі з вихідними даними, заданими в табл. 4.2.

Таблиця 4.2

Вихідні данні транспортної задачі 4.2

Постачальники A_i	Споживачі B_j				Запаси
	1	2	3	4	
1	3	5	4	6	50
2	2	4	1	5	75
3	8	3	2	4	100
4	2	5	6	1	120
Потреби	40	80	110	70	

Розв'язання. Задача оказалась відкритою, запаси постачальників (345) перевищують попит споживачів (300) на 45 одиниць, тому вводимо фіктивний пункт споживання з обсягом споживання у 45 одиниць продукції. Нові вихідні дані наведені у табл. 4.3.

Таблиця 4.3

Вихідні данні транспортної задачі з фіктивним споживачем

Постачальники A_i	Споживачі B_j					Запаси
	1	2	3	4	**5**	
1	3	5	4	6	**0**	50
2	2	4	1	5	**0**	75
3	8	3	2	4	**0**	100
4	2	5	6	1	**0**	120
Потреби	40	80	110	70	**45**	

Складемо математичну модель ТЗ і наведемо основну інтерпретацію моделі ТЗ згідно з синтаксисом середовища MATLAB.

$$f = 3\cdot x_{11} + 5\cdot x_{12} + 4\cdot x_{13} + 6\cdot x_{14} + 0\cdot x_{15}$$
$$+ 2\cdot x_{21} + 4\cdot x_{22} + 1\cdot x_{23} + 5\cdot x_{24} + 0\cdot x_{25}$$
$$+ 8\cdot x_{31} + 3\cdot x_{32} + 2\cdot x_{33} + 4\cdot x_{34} + 0\cdot x_{35}$$
$$+ 2\cdot x_{41} + 5\cdot x_{42} + 6\cdot x_{43} + 1\cdot x_{44} + 0\cdot x_{45} \to \min$$

Вектор коефіцієнтів цільової функції:

```
f=[3; 5; 4; 6; 0;
   2; 4; 1; 5; 0;
   8; 3; 2; 4; 0;
   2; 5; 6; 1; 0];
```

Від кожного постачальника можна вивести обсяг продукції не більше наявної кількості:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 50 \\ x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 75 \\ x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 100 \\ x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 120 \end{cases}$$

Потреба кожного споживача в продукції повинна бути задоволена:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 40 \\ x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 80 \\ x_{13} + x_{23} + x_{33} + x_{43} + x_{53} = 110 \\ x_{14} + x_{24} + x_{34} + x_{44} + x_{54} = 70 \\ x_{15} + x_{25} + x_{35} + x_{45} + x_{55} = 45 \end{cases}$$

$x_{ij} \ge 0, i = \overline{1,4},\ j = \overline{1,5}$

Матриця коефіцієнтів системи лінійних нерівностей:

```
Aeq=[1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 1 0 0 0 0 1 0
0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
0 1];
```

Вектор вільних членів системи лінійних нерівностей:

```
Beq = [50; 75; 100; 120;
40; 80; 110; 70; 45];
```

Вектор нижніх меж для змінних $x_{ij}, i = \overline{1,4},\ j = \overline{1,5}$:

```
lb = zeros(20, 1);
```

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `linprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
% Розв'язок транспортної задачі шляхом її приведення
% до задачі лінійного програмування
clc
clear all
% Вектор коефіцієнтів
f=[3; 5; 4; 6; 0; 2; 4; 1; 5; 0; 8; 3; 2; 4; 0; 2; 5; 6; 1; 0];
% Бінарні коефіцієнти в обмеженнях
```

```
Aeq=[1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
     0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1

     1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
     0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
     0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0
     0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0
     0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1];

% Праві частини обмежень
Beq = [50; 75; 100; 120; 40; 80; 110; 70; 45];

% Нульовий вектор для завдання умови невід'ємності
% обсягів перевезень
lb = zeros(20, 1);

% Функція розрахунку шуканих обсягів перевезень
[x fval] = linprog(f, [], [], Aeq, Beq, lb);
Xopt = reshape(x,4,5)'
Fopt = fval
```

Результати роботи програми:

```
Optimization terminated.

Xopt =

    0.0000   15.0000    0.0000    0.0000   35.0000
    0.0000    0.0000   75.0000    0.0000    0.0000
    0.0000   65.0000   35.0000    0.0000    0.0000
   40.0000    0.0000    0.0000   70.0000   10.0000

Fopt = 565.0000
```

Таким чином, мінімальні витрати складуть $f_{\min} = 565$ ум. од. відповідно до оптимального плану: $X_{opt} = \begin{pmatrix} 0 & 15 & 0 & 0 & 35 \\ 0 & 0 & 75 & 0 & 0 \\ 0 & 65 & 35 & 0 & 0 \\ 40 & 0 & 0 & 70 & 10 \end{pmatrix}$.

Слід звернути увагу, що розв'язок ТЗ з фіктивним постачальником у лабораторному практикумі не наводиться окремою задачею, тому що його програмна реалізація аналогічна прикладу 4.2 і не має практичного інтересу.

4.2. Розв'язування типових задач у середовищі MATLAB

Задача про призначення є окремим випадком ТЗ і в загальному випадку може бути сформульована таким чином. Є n працівників, які можуть виконувати n робіт, причому використання i-го працівника на j-й роботі, наприклад, приносить дохід c_{ij}. Потрібно доручити кожному з працівників виконання однієї цілком визначеної роботи, щоб максимізувати сумарний дохід. Якщо в задачі про призначення елементом матриці оцінок є, наприклад, час виконання кожним працівником будь-якої з робіт, то цільова функція цієї задачі буде мінімізуватися. Слід зауважити також, що для розв'язання задачі про призначення в програмному середовищі MATLAB використовується функція `linprog`.

Приклад 4.3 (задача про призначення). На підприємстві є 4 автомобілі та 4 маршрути, для яких отримана матриця витрат (табл. 4.4).

Таблиця 4.4

Вихідні дані задачі про призначення

Номери автомобіля	Номер маршруту			
	1	2	3	4
1	12	8	10	12
2	4	13	10	7
3	9	8	12	11
4	17	16	12	6

Потрібно призначити кожен з автомобілів на один з маршрутів таким чином, щоб всі наявні витрати були мінімальними.

Розв'язання. Введемо змінні:

$$X_{ij} = \begin{cases} 1, \textit{якщо } i-\textit{й автомобіль призначен на } j-\textit{й маршрут} \\ 0, \textit{у протилежному випадку} \end{cases}.$$

Задача полягає в тому, щоб знайти розподіл $X = (X_{ij})$ автомобілів за маршрутами (тобто знайти матрицю призначень), який мінімізує цільову функцію:

$$F = \sum_{i \in I} \sum_{j \in J} c_{ij} \cdot X_{ij} \to \min,$$

де i, I – номер і кількість автомобілів;

j, J – номер і кількість маршрутів;

c_{ij} – витрати на перевезення вантажів для i-го автомобіля на j-му маршруті.

Обмеження задачі мають вигляд:

1) на кожний заздалегідь встановлений маршрут скеровується тільки один автомобіль:

$$\sum_{i \in I} X_{ij} = 1,\ j \in J;$$

2) кожен автомобіль призначений тільки на один заздалегідь установлений маршрут:

$$\sum_{j \in J} X_{ij} = 1, i \in I.$$

Складемо математичну модель задачі про призначення та наведемо основну інтерпретацію моделі задачі про призначення згідно з синтаксисом середовища MATLAB.

$$\begin{aligned} f = {} & 12 \cdot x_{11} + 8 \cdot x_{12} + 10 \cdot x_{13} + 12 \cdot x_{14} \\ & + 4 \cdot x_{21} + 13 \cdot x_{22} + 10 \cdot x_{23} + 7 \cdot x_{24} \\ & + 9 \cdot x_{31} + 8 \cdot x_{32} + 12 \cdot x_{33} + 11 \cdot x_{34} \\ & + 17 \cdot x_{41} + 16 \cdot x_{42} + 12 \cdot x_{43} + 6 \cdot x_{44} \to \min \end{aligned}$$

Вектор коефіцієнтів цільової функції:

```
f=[12; 8; 10; 12;
   4; 13; 10; 7;
   9; 8; 12; 11;
   17; 16; 12; 6];
```

На кожний заздалегідь встановлений маршрут призначений тільки один автомобіль:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 1 \\ x_{21} + x_{22} + x_{23} + x_{24} = 1 \\ x_{31} + x_{32} + x_{33} + x_{34} = 1 \\ x_{41} + x_{42} + x_{43} + x_{44} = 1 \end{cases}$$

Матриця коефіцієнтів системи лінійних нерівностей:

```
A=[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

Кожен автомобіль призначений тільки на один заздалегідь установлений маршрут:

```
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1];
```

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ x_{14} + x_{24} + x_{34} + x_{44} = 1 \end{cases}$$

Вектор вільних членів системи лінійних нерівностей:

```
Beq = [1; 1; 1; 1; 1; 1; 1; 1];
```

Вектор нижніх меж:

```
lb = zeros(1, 16);
```

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `linprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clc
clear all
% Вектор коефіцієнтів цільової функції
f=[12; 8; 10; 12; 4; 13; 10; 7; 9; 8; 12; 11; 17; 16; 12; 6];
% Бінарні коефіцієнти в обмеженнях
d = ones(1,4);
Aeq = [blkdiag(d, d, d, d); diag(d) diag(d) diag(d) diag(d)];

% Праві частини обмежень
Beq = [ones(1, 8)];

% Вектор нижніх меж:
lb = [zeros(1,16)];

% Розв'язання задачі про призначення
[x fval] = linprog(f, [], [], Aeq, Beq, lb);
Xopt = reshape(x,4,4)
Fopt = fval
```

Результати роботи програми:

```
Optimization terminated.

Xopt =
    0.0000    1.0000    0.0000    0.0000
    0.0000    0.0000    1.0000    0.0000
    1.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    1.0000

Fopt = 28.0000
```

Отримані результати означають: $x_{21} = x_{12} = x_{33} = x_{44} = 1$. Тобто другий автомобіль слід направити на перший маршрут, перший автомобіль − на другий маршрут, третій автомобіль − на третій, четвертий автомобіль − на четвертий. Мінімальні витрати складуть 28 у. о.

Запитання для самоперевірки

1. Сформулюйте транспортну задачу.
2. Що є цільовою функцією в транспортній задачі?
3. У чому полягають обмеження транспортної задачі?
4. Яку модель транспортної задачі називають закритою, відкритою?
5. Сформулюйте умову балансу транспортної задачі.
6. Як відкриту модель транспортної задачі звести до закритої?
7. Опишіть інтерпретацію моделі транспортної задачі відносно до програмного середовища MATLAB.

Завдання до лабораторної роботи

На складах A_i, $i = \overline{1,3}$, оптової бази зосереджений однорідний товар у кількості a_i одиниць. Цей товар необхідно перевезти в чотири магазини B_j, $j = \overline{1,4}$. Кожен з магазинів повинен отримати, відповідно, b_j одиниць товару. Транспортні витрати c_{ij} на перевезення однієї одиниці товару зі складу A_i в магазин B_j відомі. Усі необхідні числові дані наведені в табл. 4.5. Знайдіть оптимальний план перевезення товару зі складів до магазинів з мінімальними сумарними витратами на перевезення.

Для цього у середовищі MATLAB виконати такі дії:

1) побудувати економіко-математичну модель задачі з перевезення товару зі складів до магазинів з мінімальними сумарними витратами на перевезення;

2) для заданої математичної моделі вихідної задачі запрограмувати у середовищі MATLAB алгоритм розв'язку останньої;

3) указати магазини, які недоотримають товар, або склади, які в повному обсязі вивезуть товар, а також його кількість.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Таблиця 4.5

Варіанти завдань до лабораторної роботи 4

Варіант 1	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	16	30	17	10	**27**
2	30	27	26	9	**20**
3	13	4	22	3	**40**
Потреби	**33**	**13**	**27**	**17**	

Варіант 2	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	5	6	5	6	**80**
2	1	4	3	2	**60**
3	2	7	1	2	**25**
Потреби	**25**	**10**	**25**	**50**	

Варіант 3	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	4	3	5	3	**60**
2	2	7	1	2	**25**
3	3	5	6	3	**50**
Потреби	**55**	**45**	**40**	**60**	

Варіант 4	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	5	6	4	3	**80**
2	1	4	4	3	**60**
3	6	7	4	3	**40**
Потреби	**35**	**20**	**40**	**45**	

Варіант 5	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	3	2	4	3	**60**
2	5	6	4	3	**80**
3	3	2	4	3	**50**
Потреби	**60**	**45**	**40**	**60**	

Варіант 6	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	5	6	4	3	**80**
2	1	4	4	3	**60**
3	6	7	4	3	**40**
Потреби	**35**	**20**	**40**	**45**	

Варіант 7	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	3	6	5	6	**35**
2	2	7	1	4	**25**
3	2	4	6	7	**95**
Потреби	**70**	**45**	**35**	**60**	

Варіант 8	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	5	4	3	6	**50**
2	2	4	7	1	**70**
3	6	4	5	2	**20**
Потреби	**25**	**10**	**40**	**30**	

Варіант 9	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	4	2	5	4	**25**
2	3	4	5	6	**20**
3	2	5	6	7	**40**
Потреби	**35**	**15**	**30**	**20**	

Варіант 10	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4	
1	5	6	3	5	**50**
2	1	4	7	1	**25**
3	6	7	5	6	**70**
Потреби	**35**	**20**	**45**	**40**	

Закінчення табл. 4.5

Варіант 11	Споживачі B_j				**Запаси**	**Варіант 12**	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4		Постачальники A_i	1	2	3	4	
1	2	4	4	3	**35**	1	3	5	5	3	**50**
2	6	4	2	7	**25**	2	4	2	1	4	**55**
3	2	4	3	5	**40**	3	3	4	6	6	**60**
Потреби	**45**	**40**	**60**	**20**		**Потреби**	**15**	**45**	**20**	**45**	
Варіант 13	Споживачі B_j				**Запаси**	**Варіант 14**	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4		Постачальники A_i	1	2	3	4	
1	3	2	4	3	**60**	1	3	5	6	3	**50**
2	5	6	4	3	**80**	2	6	2	2	7	**70**
3	3	2	4	3	**50**	3	2	3	2	4	**60**
Потреби	**60**	**45**	**40**	**60**		**Потреби**	**15**	**25**	**70**	**45**	
Варіант 15	Споживачі B_j				**Запаси**	**Варіант 16**	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4		Постачальники A_i	1	2	3	4	
1	3	6	5	6	**35**	1	5	4	3	6	**50**
2	2	7	1	4	**25**	2	2	4	7	1	**70**
3	2	4	6	7	**95**	3	6	4	5	2	**20**
Потреби	**70**	**45**	**35**	**60**		**Потреби**	**25**	**10**	**40**	**30**	
Варіант 17	Споживачі B_j				**Запаси**	**Варіант 18**	Споживачі B_j				**Запаси**
Постачальники A_i	1	2	3	4		Постачальники A_i	1	2	3	4	
1	3	5	5	3	**50**	1	2	4	4	3	**35**
2	4	2	1	4	**25**	2	6	4	2	7	**45**
3	3	4	6	6	**20**	3	2	4	3	5	**80**
Потреби	**45**	**45**	**40**	**45**		**Потреби**	**25**	**40**	**40**	**20**	

У лабораторній роботі подано опис основних команд для розв'язання транспортних задач лінійного програмування. Окремо розглянуто способи розв'язання деяких типових транспортних задач, а саме, ТЗ з фіктивним споживачем та задачі про призначення. Використовуючи розглянуті команди середовища MATLAB, можна ілюструвати розв'язання типових задач впродовж лабораторної роботи.

Лабораторна робота 5
Цілочислове програмування

Мета роботи: набуття практичних навичок для розв'язання задач цілочислового програмування та задач з бінарними (булевими) змінними за допомогою стандартних процедур середовища MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель вихідних задач.

2. За допомогою вбудованої функції `bintprog` середовища MATLAB знайти оптимальний розв'язок задачі з бінарними (булевими) змінними.

3. За допомогою вбудованої функції `intlinprog` середовища MATLAB знайти оптимальний розв'язок задачі цілочислового програмування.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

5.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

BINTPROG **Розв'язання задачі цілочислового програмування**

Необхідно мінімізувати значення виразу $f^T \cdot x$ з обмеженнями:

$$A \cdot x \le b;$$

$$Aeq \cdot x = beq;$$

де f, b, beq – вектори;

A та Aeq – матриці;

x – цілочисловий вектор розв'язку (тобто його компоненти повинні приймати значення 0 або 1).

Синтаксис

```
x = bintprog(f)
x = bintprog(f, A, b)
x = bintprog(f, A, b, Aeq, beq)
x = bintprog(f, A, b, Aeq, beq, x0)
```

```
x = bintprog(f, A, b, Aeq, beq, x0, options)
[x, fval] = bintprog(...)
[x,fval, exitflag] = bintprog(...)
[x, fval, exitflag, output] = bintprog(...)
```

Опис

`x = bintprog (f)` розв'язує задачі цілочислового програмування $f^T \cdot x \to \min$.

`x = bintprog (f, A, b)` розв'язує задачі цілочислового програмування $f^T \cdot x \to \min$ за умови $A \cdot x \le b$.

`x = bintprog (f, A, b, Aeq, beq)` розв'язує попередню задачу з додатковими умовами типу рівностей $Aeq \cdot x \le beq$.

`x = bintprog (f, A, b, Aeq, beq, x0)` установлює початкову точку пошуку `x0`. Якщо точка `x0` знаходиться в неприпустимій області, то команда `bintprog` приймає довільну початкову точку.

`x = bintprog (f, A, b, Aeq, Beq, x0, options)` – за умови оптимізації використовується опція зі структури `options`, яку можна задати за допомогою функції `optimset`.

`[X, fval] = bintprog (...)` повертає `fval` як значення цільової функції в точці `x`.

`[X, fval, exitflag] = bintprog (...)` повертає параметр `exitflag` з описом вихідних умов команди `bintprog`. Більш докладно опис даної функції і значення окремих аргументів можна знайти в табл. А.1 додатка А.

`[X, fval, exitflag, output] = bintprog (...)` повертає структуру `output`, яка містить інформацію про дані результати оптимізації.

Аргументи

Таблиця 5.1 містить загальний опис аргументів, переданих в `bintprog`.

Табл. 5.2 містить загальний опис вихідних параметрів `exitflag` та `output`, які мають ряд особливостей.

Таблиця 5.1

Вхідні аргументи функції `bintprog`

Аргумент	Опис
`A, b`	Матриця `A` і вектор `b` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних нерівностей і вектора правої частини: `A*x <= b`
`f`	Вектор коефіцієнтів лінійної цільової функції
`x0`	Початкова точка пошуку за даним алгоритмом
`Aeq, beq`	Матриця `Aeq` і вектор `beq` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних рівностей і вектора правої частини: `Aeq*x = beq`
`options`	Структура з параметрами опцій оптимізації, за допомогою якої задаються параметри опцій оптимізації. Більш детально інформацію про ці параметри можна знайти в табл. А.1 додатка А

Таблиця 5.2

Вихідні аргументи функції `bintprog`

Назва вихідного аргументу	Опис	
`exitflag`	Певне ціле число, що ідентифікує причину зупинки алгоритму. Перелік прийнятих значень і відповідних причин зупинки алгоритму:	
	1	функція зійшлася до деякого розв'язку `x`
	0	число ітерацій перевищило значення `options.MaxIter`
	-2	ця задача не має розв'язку
	-4	число перебраних вузлів перевищує значення `options.MaxNodes`
	-5	час перебору перевищує значення `options.MaxTime`
	-6	число ітерацій розв'язувача `LP` для деякого вузла для розв'язання задачі `LP`-релаксації перевищило значення `options.MaxRLP`
`output`	Структура з інформацією про результати оптимізації. Поле цієї структури має вигляд:	
	`iterations`	число виконаних ітерацій
	`nodes`	число вузлів перебору
	`time`	перевищення часу роботи алгоритму
	`algorithm`	використовуваний алгоритм
	`message`	причина зупинення роботи алгоритму

У табл. 5.3 наведений опис опцій команди `bintprog`. Для установки або зміни значень поля цієї структури використовується команда `optimset`.

Таблиця 5.3

Опції команди `bintprog`

Назви параметрів	Опис параметрів
`BranchStrategy`	Тип алгоритму, який використовується для вибору змінної розгалуження в дереві пошуку: `'Mininfeas'` – вибір змінної з мінімальною цілочисловою недосяжністю. Тобто вибираються змінні зі значенням, близьким до 0 або 1, але не рівним 0 або 1. `'Maxinfeas'` – вибір змінної з максимальною цілочисловою недосяжністю. Тобто вибираються змінні зі значенням близьким до 0,5 (що приймається за замовчуванням)
`Diagnostics`	Відображення діагностичної інформації про цю функцію
`Display`	Рівень відображення: `'Off'` немає висновку відображення; `'Iter'` відображення виводиться на кожній ітерації; `'Final'` (приймається за замовчуванням). Відображення тільки за умови остаточного виведення
`DispNodeInterval`	Інтервал відображень вузлів
`MaxIter`	Максимальне число допустимих ітерацій
`MaxNodes`	Максимальне число рішень (або вузлів) функції перебору
`MaxRLPIter`	Максимальне число ітерацій розв'язувача `LP` для деякого вузла для розв'язання задачі `LP`-релаксації
`MaxTime`	Максимальна кількість часу в секундах для виконання заданої функції
`TolCon`	Кінцева допустима межа на порушення заданого обмеження
`TolFun`	Кінцева межа для значення функції
`TolXInteger`	Допустима межа, за якої розглядаються значення змінних
`NodeSearchStrategy`	Стратегія алгоритму, який використовується для відбору наступного вузла при переборі в дереві пошуку: `'Df'` – стратегія глибини першого пошуку. На кожному вузлі дерева пошуку (якщо дочірній вузол з нижнього рівня дерева, яке раніше не було використано) в даному алгоритмі проводиться вибір одного такого елемента для перебору. В іншому випадку даний алгоритм переходить до певного вузла на один рівень вгору на дереві та вибирається дочірній вузол на один рівень вниз від даного вузла `'Bn'` – найкраща стратегія перебору вузлів, за якої відбирається вузол з найменшим граничним значенням цільової функції
`TolRLPFun`	Кінцева допустима межа на значення функції задачі релаксації лінійного програмування

Алгоритм

У функції `bintprog` для розв'язання задачі цілочислового програмування використовується алгоритм лінійного програмування (`LP`) на основі методу гілок і меж. У цьому алгоритмі проводиться перебір оптимальних розв'язків задачі цілочислового програмування шляхом розв'язання певного набору задач `LP`-релаксації, в якому вимога цілочисельності змінних замінюється на більш слабке обмеження. Даний алгоритм включає:

1) перебір цілочисельних допустимих рішень;

2) коригування найкращою цілочисленною допустимою точкою в міру просування деревом пошуку;

3) перевірка неможливості досягнення кращих цілочисельних рішень за допомогою низки завдань лінійного програмування.

INTLINPROG **Змішане цілочислове лінійне програмування**

Необхідно мінімізувати значення виразу $f^T \cdot x$ з обмеженнями:

$$A \cdot x \le b;$$
$$Aeq \cdot x = beq;$$
$$lb \le x \le ub,$$

де f, b, beq – вектори;

A та Aeq – матриці;

x – цілочисловий вектор розв'язку, тобто його компоненти повинні приймати значення 0 або 1.

Синтаксис

```
x = intlinprog(f,intcon,A,b)
x = intlinprog(f,intcon,A,b,Aeq,beq)
x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub)
x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options)
x = intlinprog(problem)
[x,fval,exitflag,output] = intlinprog(...)
```

Опис

`x = intlinprog(f,intcon,A,b)` розв'язує задачі цілочислового програмування $f^T \cdot x \to \min$, причому компоненти `x` в `intcon` є цілими числами та виконується умова $A \cdot x \le b$.

`x = intlinprog(f,intcon,A,b,Aeq,beq)` розв'язує попередню задачу цілочислового програмування $f^T \cdot x \to \min$, додатково задовольняючи обмеженням рівності $Aeq \cdot x = beq$. Необхідно встановити `A = []` та `b = []`, якщо не існує нерівностей.

`x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub)` установлює, відповідно, нижню та верхню межі змінної `x` так, що розв'язок завжди знаходиться в діапазоні $lb \le x \le ub$.

`x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options)` для оптимізації використовується опція зі структури `options`, яку можна задати за допомогою функції `optimset`.

`x = intlinprog(problem)` використовує структуру `problem`, щоб упакувати всі вхідні параметри функції `intlinprog`.

`[x,fval,exitflag,output] = intlinprog(...)` повертає `fval` як значення цільової функції в точці `x` і структуру `output`, яка містить інформацію про дані результати оптимізації.

Аргументи

Табл. 5.4 містить загальний опис аргументів, переданих в `intlinprog`.

Таблиця 5.4

Вхідні аргументи функції `intlinprog`

Аргумент	Опис
`A, b`	Матриця `A` і вектор `b` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних нерівностей і вектора правої частини: `A*x <= b`
`Aeq,beq`	Матриця `Aeq` і вектор `beq` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних рівностей і вектора правої частини: `Aeq*x = beq`
`f`	Вектор коефіцієнтів лінійної цільової функції
`Intcon`	Вектор цілочисельних обмежень
`lb, ub`	Відповідно нижні та верхні межі змінної `x`
`options`	Структура з параметрами опцій оптимізації, за допомогою якої задаються параметри опцій оптимізації. Більш детально інформацію про ці параметри можна знайти в табл. А.1 додатка А

У табл. 5.5 наведено загальний опис вихідних параметрів `exitflag` та `output`, які мають ряд особливостей.

Таблиця 5.5

Вихідні аргументи функції `intlinprog`

`exitflag`	Певне ціле число, що ідентифікує причину зупинки алгоритму. Далі наводиться перелік прийнятих значень і відповідних причин зупинки алгоритму:	
	1	функція зійшлася до деякого розв'язку `x`
	0	число ітерацій перевищило значення `options.MaxIter`
	-2	ця задача не має розв'язку
	-3	число перебраних вузлів перевищує значення `options.MaxNodes`
`output`	Структура з інформацією про результати оптимізації. Поле цієї структури має вигляд:	
	`relativegap`	відносна процентна різниця між верхніми (`U`) і нижніми (`L`) межами цільової функції, яку `intlinprog` обчислює в своєму алгоритмі розгалуження і прив'язки
	`absolutegap`	різниця між верхньою і нижньою межами цільової функції, яку `intlinprog` обчислює в своєму алгоритмі розгалуження і прив'язки
	`numfeaspoints`	число знайдених цілочислових точок
	`numnodes`	число вузлів у алгоритмі з розгалуженням і межею. Якщо зрив був виявлений під час попередньої обробки або під час початкових розрізів, то `numnodes = 0`
	`constrviolation`	порушення обмежень, яке є додатним для порушених обмежень
	`message`	причина зупинення роботи алгоритму

Таким чином, розв'язання задач лінійного програмування у середовищі MATLAB, які потребують цілочислового розв'язку, можливо здійснити за рахунок вбудованої функції `bintprog`, використовуючи певні налаштування з табл. 5.1 – 5.5.

5.2. Розв'язування типових задач у середовищі MATLAB

Приклад 5.1. Знайти максимум функції

$$f = 23x_1 + 41x_2 + 18x_3 + 25x_4 + 30x_5$$

за умов

$$\begin{cases} 4x_1 + x_2 + 2x_3 + x_4 + 2x_5 = 4 \\ 7x_1 + 8x_2 + 6x_3 \le 15 \\ 11x_2 + 5x_3 + 12x_4 + 9x_5 \le 27; \end{cases}$$

$$x_j \in \{0,1\},\ j = \overline{1,5}.$$

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `bintprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clear all
close all
clc % Видаляються всі поточні змінні з пам'яті MATLAB,
    % закриваються всі графічні вікна, очищається екран
    % консолі;
f=[-23;  -41;  -18;  -25;  -30];% Вектор коефіцієнтів цільової
функції;
%Матриці коефіцієнтів системи лінійних нерівностей:
A = [7 8 6 0 0; 0 11 5 12 9];
Aeq = [4 1 2 1 2];
% Вектор вільних членів:
b = [15; 27];
beq = [4];% Вектор вільних членів;
% Звернення до програми лінійного програмування:
[x,fval,exitflag,output] = bintprog(f,A,b,Aeq,beq)
```

Результати роботи програми:

```
Optimization terminated.

x =
     0
     0
     1
     0
     1

fval = 48

exitflag = 1
output =

          iterations: 26
               nodes: 27
                time: 0.2964
           algorithm: 'LP-based branch-and-bound'
      branchStrategy: 'maximum integer infeasibility'
    nodeSrchStrategy: 'best node search'
             message: 'Optimization terminated.'
```

Звіт показує, що розв'язок знайдено за 25 ітерацій алгоритмом гілок і меж, заснованим на розв'язанні LP-задач. Для цього переглянуто 27 вузлів, використані стратегії розгалуження maxinfeas і вибору вузлів bn.

Розглянемо приклад використання цілочислового програмування до розв'язку економічних задач.

Приклад 5.2. Для придбання обладнання, що розміщується на виробничій площі 38 м^2, фірма виділяє 20 млн ум. од. Є одиниці обладнання двох типів: типу А вартістю 5 млн. ум. од., що потребує виробничої площі 8 м^2 і має продуктивність 7 тис. од. продукції за зміну; типу Б − вартістю 2 млн ум. од., що займає площу 4 м^2 і дає за зміну 3 тис. од. продукції. Потрібно розрахувати оптимальний варіант придбання обладнання, що забезпечує максимум продуктивності ділянки.

Розв'язання. Математична модель задачі може бути записана таким чином:

$$f = 7x_1 + 3x_2 \to \max$$
$$\begin{cases} 5x_1 + 2x_2 \le 20 \\ 8x_1 + 4x_2 \le 38 \end{cases},$$
$$x_1, x_2 \ge 0,\ x_1, x_2 \in Int$$

де x_1, x_2 – кількість придбаних машин типу А і типу Б, відповідно.

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції intlinprog, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clear all
close all
clc % Видаляються всі поточні змінні з пам'яті MATLAB,
    % закриваються всі графічні вікна, очищається екран
    % консолі;
f=[-7; -3];% Вектор коефіцієнтів цільової функції;
%Матриці коефіцієнтів системи лінійних нерівностей:
A = [5 2; 8 4];
% Вектор вільних членів:
b = [20; 38];
intcon = [1 2];
lb = zeros(2,1);
% Звернення до програми лінійного програмування:
[x,fval,exitflag,output] = intlinprog(f,intcon,A,b,[],[],lb);
x
f=-fval
exitflag
output
```

Результати роботи програми:

```
Optimal solution found.
x =
    2.0000
    5.0000

f = 29.0000

exitflag = 1

output =
        relativegap: 0
        absolutegap: 0
      numfeaspoints: 1
           numnodes: 0
    constrviolation: 0
            message: 'Optimal solution found.…'
```

Таким чином, придбання двох машин типу А і п'яти машин типу Б забезпечує максимальну продуктивність ділянки, у 29 тис. од. продукції в зміну.

Приклад 5.3 (задача комівояжера). У загальному випадку ця задача може бути сформульована таким чином. Існує n міст, пронумерованих числами від 1 до n. Комівояжер, виїжджаючи з міста 1, повинен побувати в кожному місті рівно один раз і повернутися в початковий пункт. Нехай відомі відстані c_{ij} між містами $(i, j = \overline{1, n})$. Потрібно знайти найкоротший маршрут.

Отже, нехай існує 6 пунктів, які повинен відвідати комівояжер одноразово, мінімізуючи пройдений шлях, і повернутися в початкове місто. Відстані між містами задані матрицею

$$C = \begin{pmatrix} \infty & 27 & 43 & 16 & 30 & 26 \\ 7 & \infty & 16 & 1 & 30 & 25 \\ 20 & 13 & \infty & 35 & 5 & 0 \\ 21 & 16 & 25 & \infty & 18 & 18 \\ 12 & 46 & 27 & 48 & \infty & 5 \\ 23 & 5 & 5 & 9 & 5 & \infty \end{pmatrix}.$$

Розв'язання. Запишемо математичну модель задачі:

$$\begin{cases} 1, \textit{ якщо комівояжер переїжджає з міста } i \textit{ до міста } j, \\ 0, \textit{ в протилежному випадку.} \end{cases}$$

Потрібно мінімізувати $F(x)=\sum_{i=1}^{6}\sum_{j=1}^{6} c_{ij}\cdot x_{ij}$ з обмеженнями:

$$\begin{cases} \sum_{i=1}^{6} x_{ij} = 1 \\ \sum_{j=1}^{6} x_{ij} = 1 \\ u_i - u_j + 6x_{ij} \le 5. \end{cases}.$$

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `intlinprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clc
clear all
f = [inf; 27; 43; 16; 30; 26; 7; inf; 16; 1; 30; 25; 20; 13; inf;
35; 5; 0; 21; 16; 25; inf; 18; 18; 12; 46; 27; 48; inf; 5; 23; 5;
5; 9; 5; inf];
Aeq = [1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;
       0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1];
beq = [1; 1; 1; 1; 1; 1];
lb = zeros(36,1);
intcon = [1 2 3 4 5 6]
[x,fval] = intlinprog(f, intcon, [], [], Aeq, beq, lb)
```

Результати роботи програми :

```
Optimal solution found.
x =
0.0000   0.0000   0.0000   1.0000   0.0000   0.0000
1.0000   0.0000   0.0000   0.0000   0.0000   0.0000
0.0000   0.0000   0.0000   0.0000   1.0000   0.0000
0.0000   0.0000   1.0000   0.0000   0.0000   0.0000
0.0000   0.0000   0.0000   0.0000   0.0000   1.0000
0.0000   1.0000   0.0000   0.0000   0.0000   0.0000
```

```
Fopt = 63.0000

exitflag = 1
output =
        relativegap: 0
        absolutegap: 0
      numfeaspoints: 1
           numnodes: 0
    constrviolation: 0
            message: 'Optimal solution found.…'
```

Отже, маємо маршрут (1,4) − (2,1) − (3,5) − (4,3) − (5,6) −(6,2), або 1 $\rightarrow$ $\rightarrow$ 4 $\rightarrow$ 3 $\rightarrow$ 5 $\rightarrow$ 6 $\rightarrow$ 2 $\rightarrow$ 1, довжина якого (значення цільової функції) дорівнює 63.

Запитання для самоперевірки

1. Сформулюйте задачу цілочисельного лінійного програмування.

2. Які функції програмного середовища MATLAB застосовуються до їх розв'язання?

3. Навести алгоритм методу гілок і меж для розв'язання задач цілочисельного ЛП.

4. Яке призначення функції `intlinprog`?

5. Назвіть вхідні та вихідні параметри функції `intlinprog`.

6. Яке призначення функції `bintprog`?

7. Назвіть вхідні та вихідні параметри функції `bintprog`.

8. У чому полягає відмінність функцій `bintprog` та `intlinprog`?

9. Назвіть оптимізаційні задачі, що зводяться до задач лінійного цілочисельного програмування.

Завдання до лабораторної роботи

Для розв'язання задач цілочисельного ЛП необхідно:

1) для заданих математичних моделей вихідних задач запрограмувати у середовищі MATLAB алгоритм розв'язку останніх;

2) за допомогою вбудованої функції `optimset` зробити відповідні налаштування функції `intlinprog` середовища MATLAB таким чином, щоб знайдений оптимальний розв'язок задачі був повністю цілочисельним;

3) за допомогою вхідного параметра `options` установити додаткові налаштування функції `intlinprog`, вибрати різні числа допустимих ітерацій, провести порівняльний аналіз отриманих результатів.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Варіанти завдань наведені у табл. 5.6.

Таблиця 5.6

Варіанти завдань до лабораторної роботи 5

№ варіанта та математична модель ЗЛП	№ варіанта та математична модель ЗЛП
1	2
$z = x_1 - x_2 + x_3 + x_4 \to \max$ 1. $\begin{cases} 5x_1 - 3x_2 + x_3 - 2x_4 = 1 \\ 9x_1 - 4x_2 + 2x_3 - 3x_4 = 6 \\ 2x_1 + 3x_2 + x_3 \ge 0 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$	$z = 5x_1 + x_2 - 7x_3 \to \min$ 2. $\begin{cases} x_1 + x_2 - 2x_3 \le 3 \\ 2x_1 + x_2 - 3x_3 \le 7 \\ 3x_1 - 7x_3 \le 10 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = 2x_1 + x_2 - x_3 \to \min$ 3. $\begin{cases} 2x_1 + x_2 - x_3 \ge 5 \\ x_1 + 2x_2 + x_3 \le 7 \\ x_1 - x_2 + x_3 \ge 1 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 3x_1 + 2x_2 + x_3 \to \max$ 4. $\begin{cases} x_1 + x_2 - 2x_3 \ge 2 \\ -3x_1 + 2x_2 + x_3 \le 4 \\ x_1 + 2x_3 = 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = 3x_1 + 2x_2 + x_3 + x_4 \to \max$ 5. $\begin{cases} x_1 + 2x_2 - 2x_3 + x_4 = 2 \\ -x_1 + x_3 + x_4 = 1 \\ -3x_1 + x_2 - 2x_3 + 4x_4 = 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$	$z = x_1 + 4x_2 + x_3 \to \max$ 6. $\begin{cases} -x_1 + 2x_2 + x_3 \le 4 \\ 3x_1 + x_2 + 2x_3 \le 9 \\ 2x_1 + 3x_3 + x_3 \ge 6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = 2x_1 + x_2 + 2x_3 \to \min$ 7. $\begin{cases} x_1 + 2x_2 - x_3 \ge 2 \\ -2x_1 + x_2 + 2x_3 = 2 \\ 2x_1 + x_2 - 2x_3 \le 6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 2x_1 - x_2 + x_3 + x_4 \to \min$ 8. $\begin{cases} 5x_1 - 3x_2 - x_3 - 2x_4 = 1 \\ -x_1 + 4x_2 + 2x_3 + x_4 = 6 \\ -3x_1 + x_2 - 2x_3 + 4x_4 \ge 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$

Закінчення табл. 5.6

1	2
$z = x_1 - x_2 + x_3 + x_4 \to \max$ 9. $\begin{cases} 5x_1 - 3x_2 + x_3 - 2x_4 = 1 \\ 9x_1 - 4x_2 + 2x_3 - 3x_4 = 6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$	$z = 2x_1 - x_2 + x_3 + x_4 \to \min$ 10. $\begin{cases} 5x_1 - 3x_2 - x_3 - 2x_4 = 1 \\ -x_1 + 4x_2 + 2x_3 + x_4 = 6 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$
$z = x_1 - x_2 + x_3 + x_4 \to \max$ 11. $\begin{cases} 5x_1 - 3x_2 + x_3 - 2x_4 = 1 \\ 9x_1 - 4x_2 + 2x_3 - 3x_4 = 6 \\ 2x_1 + 3x_2 + x_3 \ge 0 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$	$z = 5x_1 + x_2 - 7x_3 \to \min$ 12. $\begin{cases} x_1 + x_2 - 2x_3 \le 3 \\ 2x_1 + x_2 - 3x_3 \le 7 \\ 3x_1 - 7x_3 \le 10 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 + x_2 + x_3 \to \min$ 13. $\begin{cases} x_1 + 2x_2 - x_3 \le 5 \\ -2x_1 + x_2 + 2x_3 = 2 \\ 2x_1 + x_2 - 2x_3 \ge 3 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = 2x_1 - x_2 - 2x_3 \to \min$ 14. $\begin{cases} x_1 + 2x_2 - x_3 \ge 1 \\ 2x_1 - x_2 - 3x_3 = 2 \\ 2x_1 + x_2 - 2x_3 \le 7 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 - x_2 + x_3 + x_4 \to \max$ 15. $\begin{cases} x_1 + x_2 + x_3 - 4x_4 \le 1 \\ x_1 - x_2 + 2x_3 - x_4 = 6 \\ 2x_1 + 3x_2 + x_3 + x_4 \ge 0 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$	$z = x_1 - x_2 + x_3 + x_4 \to \max$ 16. $\begin{cases} x_1 - 3x_2 + 2x_3 - 2x_4 \ge 1 \\ x_1 - x_2 + 2x_3 - 3x_4 \le 6 \\ x_1 + x_2 + 3x_3 = 0 \end{cases}$ $x_j \ge 0,\ j = \overline{1,4}$
$z = x_1 + x_2 + x_3 \to \min$ 17. $\begin{cases} x_1 + 2x_2 - x_3 \le 5 \\ -2x_1 + x_2 + 2x_3 = 2 \\ 2x_1 + x_2 - 2x_3 \ge 3 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + x_2 + x_3 \to \min$ 18. $\begin{cases} x_1 + 2x_2 - x_3 \le 5 \\ -2x_1 + x_2 + 2x_3 = 2 \\ 2x_1 + x_2 - 2x_3 \ge 3 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$
$z = x_1 + 3x_2 + 4x_3 \to \max$ 19. $\begin{cases} x_1 + 2x_2 + 3x_3 \le 5 \\ x_1 + x_2 + x_3 = 2 \\ 2x_1 + x_2 - 4x_3 \ge 3 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$	$z = x_1 + 5x_2 + x_3 \to \min$ 20. $\begin{cases} x_1 + 2x_2 - x_3 \le 4 \\ -2x_1 + x_2 + 2x_3 = 1 \\ 2x_1 + x_2 + x_3 \ge 2 \end{cases}$ $x_j \ge 0,\ j = \overline{1,3}$

Лабораторна робота 6
Нелінійні оптимізаційні моделі економічних систем

Мета роботи: набуття практичних навичок для розв'язання задач нелінійного програмування та задачі про формування інвестиційного портфеля як задачі квадратичного програмування за допомогою вбудованих функцій середовища MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель вихідних задач.

2. За допомогою вбудованої функції `quadprog` середовища MATLAB знайти оптимальний розв'язок задач нелінійного програмування.

3. За допомогою вбудованої функції `quadprog` середовища MATLAB знайти оптимальний розв'язок задачі про формування інвестиційного портфеля як задачі квадратичного програмування.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

6.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

QUADPROG

Розв'язати квадратичну задачу математичного програмування

Необхідно мінімізувати значення виразу $\frac{1}{2}x^T H \cdot x + f^T x$ з обмеженнями:

$$A \cdot x \le b;$$
$$Aeq \cdot x = beq;$$
$$lb \le x \le ub,$$

де f, b, beq, lb, ub – вектори;

H, A та Aeq – матриці;

x – вектор розв'язку.

Синтаксис

```
x = quadprog(H,f,A,b)
x = quadprog(H,f,A,b,Aeq,beq)
```

```
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
x = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0,
options, p1,p2,...)
[x,fval] = quadprog(...)
[x,fval,exitflag] = quadprog(...)
[x,fval,exitflag,output] = quadprog(...)
[x,fval,exitflag,output,lambda] = quadprog(...)
```

Опис

`x = quadprog(H,f,A,b)` повертає вектор `x`, який мінімізує $\frac{1}{2}x^T H \cdot x + f^T x$ за умови $A \cdot x \le b$.

`x = quadprog(H,f,A,b,Aeq,beq)` розв'язує квадратичну задачу з додатковим виконанням обмежень типу рівності $Aeq \cdot x = beq$.

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub)` визначає набір нижніх і верхніх меж для змінної `x` так, щоб розв'язок знаходився в діапазоні $b \le x \le ub$.

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)` установлює початкову точку як `x0`.

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)` проводить оптимізацію зі встановленими у структурній опції `options` параметрами оптимізації.

`x = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options, p1,p2,...)` передає параметри `p1`, `p2`, ... у функцію множників матриці Гессе, визначену за допомогою параметра `HessMult` у даній структурі опцій.

`[x,fval] = quadprog(...)` повертає значення цільової функції від `x`: `fval = 0.5 * x '* H * x + f' * x`.

`[x,fval,exitflag] = quadprog(...)` додатково повертає значення `exitflag`, який описує вихідні умови в `quadprog`.

`[x,fval,exitflag,output] = quadprog(...)` додатково повертає структурний вихід `output` з інформацією про оптимізацію.

`[x,fval,exitflag,output,lambda] = quadprog(...)` повертає структурну `lambda` з полями, що містять множники Лагранжа у вигляді розв'язку від `x`.

Аргументи

Табл. 6.1 містить загальний опис аргументів, переданих в `quadprog`.

Таблиця 6.1

Вхідні аргументи функції `quadprog`

Аргументи	Опис
`A, b`	Матриця `A` і вектор `b` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних нерівностей і вектора правої частини: `A*x <= b`
`Aeq,beq`	Матриця `Aeq` і вектор `beq` є, відповідно, коефіцієнтами матриці обмежень у вигляді лінійних рівностей і вектора правої частини: `Aeq*x = beq`
`H`	Матриця коефіцієнтів квадратичної частини цільової функції. Якщо матриця `H` несиметрична, то *MATLAB* замінює її на $(H + H^T) / 2$ (проте значення цільової функції не змінюється)
`f`	Вектор коефіцієнтів лінійної цільової функції
`lb, ub`	Відповідно, нижні та верхні межі змінної `x`
`options`	Структура з параметрами опцій оптимізації, за допомогою якої задаються параметри опцій оптимізації. Більш детально інформацію про ці параметри можна знайти в табл. А.1 додатка А

У табл. 6.2 наведено загальний опис вихідних параметрів `exitflag` та `output`, які мають ряд особливостей.

Таблиця 6.2

Вихідні аргументи функції `quadprog`

`exitflag`	Певне ціле число, що ідентифікує причину зупинення алгоритму. Перелік прийнятих значень і відповідних причин зупинення алгоритму:	
	> 0	функція зійшлася до деякого розв'язку `x`
	0	максимальне число оцінки функції або ітерацій було перевищено
	< 0	ця задача не має розв'язку
`lamda`	Структура, яка містить множники Лагранжа для розв'язання за `x` (розподіленому за типами станів). Поле структури:	
	`Lower`	нижні межі `lb`
	`Upper`	верхні межі `ub`
	`Ineqlin`	лінійні нерівності
	`Eqlin`	лінійні рівності

output	Структура з інформацією про результати оптимізації. Поля цієї структури мають вигляд:	
	Iterations	число виконаних ітерацій
	Algorithm	використовуваний алгоритм
	Cgiterations	число PCG ітерацій (тільки для крупномасштабного алгоритму)
	Firstorderopt	вимірювання оптимальності першого порядку (тільки для крупномасштабного алгоритму). Для крупно-масштабних завдань з граничними обмеженнями оптимальністю першого порядку буде нескінченна норма `v.*G`, де `v` визначається як обмеження боксу і `g` - градієнт. Для крупномасштабних завдань з тільки лінійними обмеженнями оптимальністю першого порядку буде 2 норми нормованої нев'язки (`z = M \ r`) для розрахунку скороченої сполученого градієнта з попередньою обробкою даних

Опції

`LargeScale` – у разі установки `'on'` використовується, якщо це можливо, крупномасштабний алгоритм. Для використання середньомасштабного алгоритму встановлюється `'off'`. `'On'` потребує менше часу. Якщо в задачі є тільки верхні та нижні межі, тобто відсутні лінійні рівності та нерівності, то за замовчуванням приймається крупномасштабний алгоритм. Або, якщо в задачі для `quadprog` задані тільки лінійні рівності, тобто верхньої і нижньої меж, і не специфіковані лінійні нерівності, а число рівності не більше довжини `x`, то за замовчуванням даний алгоритм використовує крупно-масштабний алгоритм. В іншому випадку використовується середньомасштабний алгоритм.

`Diagnostics` – проводиться друкування діагностичної інформації про функції, що мінімізуються.

`Display` – рівень відображення. `'Off'` відображення не відбувається, `'iter'` відображення проводиться на кожній ітерації, `'final'` – (приймається за замовчуванням) відображення тільки кінцевої інформації.

`MaxIter` – максимальне число допустимих ітерацій.

`HessMult` – максимальне число допустимих ітерацій. Покажчик функції для функції множників матриці Гессе. У разі крупномасштабної задачі ця функція обчислює добуток матриць Гессе `H*Y` без дійсного формування `H`.

Така функція має форму:

```
W = hmfun (Hinfo, Y, p1, p2, ...),
```

де `Hinfo` і додаткові параметри `p1`, `p2`, ... включають використовувані для розрахунку `H*Y` матриці. `Hinfo` має бути таким же, що і перший аргумент в `quadprog`, а `p1`, `p2`, ... є ті ж додаткові параметри, що передаються в `quadprog`.

`Y` є матриця з тим же самим числом рядків, що і розмірність даної задачі. `W = H*Y`, хоча `H` явно не формується. `quadprog` використовує `Hinfo` для розрахунку попередніх даних.

`MaxPCGIter` – максимальне число PCG (попередньо пов'язаний градієнт) ітерацій.

`PrecondBandWidth` – верхня смуга попередньої обробки для PCG. За замовчуванням використовується діагональна початкова підготовка (верхня смуга з 0). Для деяких завдань збільшення смуги знижує число ітерацій PCG.

`TolFun` – кінцева допустима точність значення функції. `TolFun` використовується в якості вихідного критерію для задач з простими верхніми і нижніми межами (`lb`, `ub`).

`TolPCG` – кінцева допустима кількість ітерацій PCG. `TolPCG` використовується в якості вихідного критерію для задач, коли є тільки обмеження у вигляді рівностей (`Aeq`, `beq`).

`TolX` – кінцеве допустиме відхилення за значенням `x`.

`TypicalX` – типові значення `x`.

6.2. Розв'язування типових задач у середовищі MATLAB

Приклад 6.1. Розв'язати задачу квадратичного програмування:

$$f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2\to\min,$$

за умови, що

$$\begin{cases} x_1+x_2\le 2 \\ -x_1+2x_2\le 2 \\ 2x_1+x_2\le 3; \end{cases}$$
$$x_1\ge 0,\ x_2\ge 0.$$

Розв'язання. Спочатку зазначимо, що дана функція в матричному запису матиме такий вигляд:

$$f(x)=\frac{1}{2}x^T Hx+f^T x=\frac{1}{2}\begin{pmatrix}x_1 & x_2\end{pmatrix}\begin{pmatrix}h_{11} & h_{12}\\ h_{21} & h_{22}\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\begin{pmatrix}c_1 & c_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=$$

$$=\frac{1}{2}\left(h_{11}x_1^2+h_{12}x_1x_2+h_{21}x_1x_2+h_{22}x_2^2\right)+c_1x_1+c_2x_2,$$

де $H=\begin{pmatrix}1 & -2\\ -2 & 2\end{pmatrix}$, $f=\begin{pmatrix}-2\\ -6\end{pmatrix}$, $x=\begin{pmatrix}x_1\\ x_2\end{pmatrix}$.

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `quadprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clc
clear all
% Матриця коефіцієнтів квадратичної частини
% цільової функції:
H = [1 -2; -2 2];
% Вектор коефіцієнтів лінійної цільової функції:
f = [-2; -6];
% Коефіцієнти обмежень:
A = [1 1; -1 2; 2 1];
b = [2; 2; 3];
lb = zeros(2,1);
% Звертання до функції:
[x, fval, exitflag, output, lamda] = quadprog(H, f, A, b, [],[], lb)
```

Результати роботи програми:

```
Optimization terminated.

x =
    0.6667
    1.3333

fval = -9.1111

exitflag = 1
```

```
output =
       iterations: 3
        algorithm: 'medium-scale: active-set'
          message: 'Optimization terminated.'
```

Після третьої ітерації (`iterations = 3`) був отриманий оптимальний розв'язок задачі квадратичного програмування: $x_1 = 0.6667, x_2 = 1.3333$, $f_{\min} = -9.1111$, який задовольняє заданій системі обмежень:

```
>> lamda.ineqlin

ans =

    4.2222
    0.2222
         0
```

Приклад 6.2. Розв'язати задачу квадратичного програмування:

$$f(x) = x_1^2 + x_2^2 + x_3^2 - 2x_1x_2 - x_1 - x_2 + x_3 \to \inf,$$

за умови, що

$$\begin{cases} x_1 + x_2 + x_3 \geq 1 \\ 2x_1 + x_2 - x_3 \geq -1, \\ x_1 - x_2 + x_3 = 0 \end{cases}$$
$$0 \leq x_i \leq 1, i = \overline{1,3}.$$

Розв'язання. На відміну від попереднього прикладу, де детально розглянуто формування вхідних даних, у розв'язанні цієї задачі буде приділено увагу вихідним даним. Отже, дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції `quadprog`, наведені у відповідній програмі (m-файл), яка виглядає так:

```
clc
clear all
% Матриця коефіцієнтів квадратичної частини
% цільової функції:
H = [2 -2 0; -2 2 0; 0 0 2];
% Вектор коефіцієнтів лінійної цільової функції:
f = [-1; -1; 1];
```

```
% Коефіцієнти обмежень:
A = [-1 -1 -1; -2 -1 1];
Aeq = [1 -1 1];
b = [-1; 1];
beq = [0];
lb = zeros(3,1);
ub = ones(3, 1);
% Звертання до функції:
[x,fval] = quadprog(H,f,A,b,Aeq,beq,lb,ub)
```

Результати роботи програми:

```
Warning: Large-scale method does not currently solve this
problem formulation,
switching to medium-scale method.

Optimization terminated.

x =
    1.0000
    1.0000
    0.0000

fval = -2.0000
```

Попередження, що міститься в перших двох рядках необхідно пояснити. *MATLAB* розв'язує задачу квадратичного програмування двома способами: алгоритмом внутрішньої точки (`Large-Scale Algorithm`) і методом перебору граней (`Medium-Scale Algorithm`). За замовчуванням використовується алгоритм внутрішньої точки. Щоб вибрати метод перебору граней, потрібно написати:

```
options = optimset ( 'LargeScale', 'off');
[X, fval] = quadprog (H, f, A, b, Aeq, beq, lb, ub, [],
options).
```

Зауважимо, що MATLAB використовує алгоритм внутрішньої точки для задач двох типів: якщо є тільки обмеження-рівності та матриця обмежень рівності має повний ранг або якщо є тільки нижні `lb` і верхні `ub` межі. Задачі, що не належать до цих двох типів, MATLAB розв'язує методом перебору граней, видаючи попередження.

MATLAB дозволяє виводити інформацію про те, як завершився розв'язок задачі. За це відповідає параметр `exitflag`. Якщо значення `exitflag` дорівнює 1, то знайдений розв'язок задачі; якщо дорівнює 0, то перевищена допустима кількість ітерацій; якщо дорівнює -2, то безліч планів задачі порожня; якщо -3, то цільова функція не обмежена знизу на безлічі планів. Інтерпретація інших значень параметра `exitflag` приведена у табл. 6.2.

Для методу перебору граней допустиме число ітерацій (`MaxIter`) за замовчуванням дорівнює 200. Значення `MaxIter` можна змінити. Щоб встановити допустиму кількість ітерацій дрівненою, наприклад, 10, потрібно написати:

```
options = optimset ('LargeScale', 'off', 'MaxIter', 10);
[X, fval] = quadprog (H, f, A, b, Aeq, beq, lb, ub, [],
options).
```

Якщо після виконання десятої ітерації рішення не буде знайдено, параметр `exitflag` стане нульовим, і на екрані з'явиться повідомлення:

```
Maximum number of iterations exceeded;
increase options.MaxIter.
```

Параметр `output` містить інформацію про процес оптимізації, число ітерацій (`iterations`) і використовуваний алгоритм (`algorithm`).

Інші поля параметра `output` описані в табл. 6.2. Запустимо з даними з прикладу 6.2 програму:

```
options = optimset ( 'LargeScale', 'off');
[X, fval, exitflag, output] = quadprog (H, f, A, b, Aeq, beq, lb,
ub, [], options);
exitflag
output.iterations
output.algorithm
```

На виході отримаємо:

```
Optimization terminated.
exitflag =
1
ans =
3
ans =
medium scale: active-set
```

Це означає, що метод перебору граней успішно завершив роботу. Для знаходження розв'язку було потрібно три ітерації.

Приклад 6.3 (задача про формування інвестиційного портфеля). На прикладі моделі Марковіца оптимізації портфеля цінних паперів розглянуто особливості використання функції `quadprog` квадратичного програмування середовища MATLAB.

Розв'язання. Гаррі Марковіц сформулював задачу оптимізації портфеля цінних паперів як задачу квадратичного програмування:

$$F = x^T \sum x \to \min, \tag{6.1}$$

$$\bar{p}^T x \geq r_{\min}, \tag{6.2}$$

$$\sum_{i=1}^{n} x_i \leq B, \tag{6.3}$$

$$x_i \geq 0, i = \overline{1,n}. \tag{6.4}$$

У задачі мінімізується ризик портфеля, який визначається дорівненим до варіації портфеля (6.1), з гарантованим середнім поверненням $r_{\min}$ (6.2) і виконанням бюджетного обмеження (6.3). У базовій моделі короткі позиції не допускаються (6.4).

У середовищі MATLAB розв'яжемо задачу квадратичного програмування за допомогою функції `quadprog`.

На виході функція `quadprog` видає оптимальний план *x* задачі квадратичного програмування і екстремальне значення цільової функції *fval*. Якщо матриця *H* несиметрична, то MATLAB замінює її на $\left(H + H^T\right)/2$ (при цьому значення цільової функції не змінюється).

Як приклад використання функції `quadprog` розглянемо портфель з трьох активів, який був сформований за вихідними даними І. Ю. Ждановим. Складемо задачу квадратичного програмування:

$$\begin{aligned} & F = 0.04x_1^2 + 0.01x_2^2 + 0.0025x_3^2 + \\ & + 0.0012x_1x_2 - 0.0008x_1x_3 \to \min \\ & 1.12x_1 + 1.1x_2 + 1.07x_3 \geq r_{\min}, \\ & x_1 + x_2 + x_3 = B, \\ & x_i \geq 0, i = \overline{1,3}. \end{aligned} \tag{6.5}$$

Спочатку необхідно перетворити вихідні дані задачі до стандартного вигляду середовища MATLAB. Вектор-стовпець плану x має вигляд $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, тоді матрицю H знаходимо з рівняння:

$$H = \frac{1}{2}\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \frac{1}{2}\begin{pmatrix} h_{11}x_1^2 + h_{21}x_1x_2 + h_{31}x_1x_3 + \\ h_{12}x_1x_2 + h_{22}x_2^2 + h_{32}x_2x_3 + \\ h_{13}x_1x_3 + h_{23}x_2x_3 + h_{33}x_3^2 \end{pmatrix} = F^T \cdot x.$$

Отже, матриця H дорівнює $H = \begin{bmatrix} 0.08 & 0.0012 & -0.0008 \\ 0.0012 & 0.02 & 0 \\ -0.0008 & 0 & 0.005 \end{bmatrix}$.

Коефіцієнти лінійної частини цільової функції заповнять вектор-рядок масиву $f = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$.

Зауважимо, що вважаючи $B = 1$, ми обчислимо частку x_i активів, вкладених у i-й актив ($i = \overline{1,3}$).

Вирішимо в MATLAB задачу квадратичного програмування. Відповідна програма (m-файл) виглядає так:

```
clc
clear all
H=[0.08 0.0012 -0.0008;0.0012 0.02 0;-0.0008 0 0.005];
f = [0 0 0];
A = [-1.12 -1.1 -1.07];
b = [-0.05];
Aeq = [1 1 1];
beq = [1];
lb = zeros(3,1);
[x,fval,exitflag,output]=quadprog(H,f,A,b,Aeq,beq,lb);
x
fval
exitflag
output.iterations
output.algorithm
```

Необхідно звернути увагу, що в програмі елементи матриць A та b взяті з протилежними знаками. Знаки "-" з'являються, оскільки обмеження-нерівності $A \cdot x \geq b$ приводяться до вигляду $-A \cdot x \leq -b$.

Запустивши програму, отримаємо:

```
Optimization terminated.
x =
    0.0520
    0.1854
    0.7626
fval =
    0.0019
exitflag = 1
ans = 1
ans =
medium-scale: active-set
```

Отже, з гарантованим середнім поверненням $r_{\min} = 5\%$ частка першого активу становить 0,052, частка другого дорівнює 0,185 і частка третього активу становить 0,763. Загальний ризик склав 19 %.

Запитання для самоперевірки

1. Сформулюйте задачу квадратичного програмування.
2. Яка функція програмного середовища MATLAB застосовується до її розв'язання?
3. Навести алгоритм розв'язання задач квадратичного програмування.
4. Яке призначення функції `quadprog`?
5. Назвіть вхідні параметри функції `quadprog`.
6. Назвіть вихідні параметри функції `quadprog`.
7. Назвіть оптимізаційні задачі, що зводяться до задач квадратичного програмування.

Завдання до лабораторної роботи

Для розв'язання задач квадратичного програмування необхідно:

1) за допомогою вбудованої функції `quadprog` середовища MATLAB знайти оптимальний розв'язок задач нелінійного програмування;

2) змінити алгоритм внутрішньої точки на метод перебору граней за допомогою функції `optimset`;

3) провести порівняльний аналіз отриманих результатів. Зробити висновки.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Варіанти завдань наведені у табл. 6.3.

Таблиця 6.3

Варіанти завдань до лабораторної роботи 6

№ варіанта	Математична модель задачі квадратичного програмування
1	2
1	$f(x)=x_1^2+x_2^2-10x_1-15x_2\to\min$ $\begin{cases}2x_1+3x_2\le 13\\ 2x_1+x_2\le 10\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
2	$f(x)=x_1^2+x_2^2-20x_1-30x_2\to\min$ $\begin{cases}5x_1+13x_2\le 51\\ 15x_1+7x_2\le 107\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
3	$f(x)=x_1^2-2x_1-x_2\to\min$ $\begin{cases}2x_1+3x_2\le 6\\ 2x_1+x_2\le 4\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
4	$f(x)=x_1^2+x_2^2-10x_1-20x_2\to\min$ $\begin{cases}9x_1+8x_2\le 72\\ x_1+2x_2\le 10\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
5	$f(x)=x_1^2+x_2^2-10x_1-8x_2\to\min$ $\begin{cases}x_1+x_2\le 12\\ x_1+4x_2\le 10\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
6	$f(x)=x_1^2+x_2^2+3x_1x_2-4x_1+x_2\to\min$ $\begin{cases}5x_1+6x_2\le 27\\ 3x_1+4x_2\le 0\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$

Продовження табл. 6.3

1	2
7	$f(x)=x_2^2-x_1x_2-x_1+3x_2\to\min$ $\begin{cases} x_1+x_2\le 1 \\ x_1-7x_2\le 5 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
8	$f(x)=x_1^2+x_2^2+3x_1x_2-4x_1+x_2\to\min$ $\begin{cases} 5x_1+6x_2\le 27 \\ 3x_1+4x_2\le 0 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
9	$f(x)=2x_1^2+3x_2^2-x_1x_2-x_1+2x_2\to\min$ $\begin{cases} 4x_1+2x_2\le 17 \\ 13x_1+21x_2\le 10 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
10	$f(x)=x_1^2-x_2^2-3x_1-4x_2\to\min$ $\begin{cases} 5x_1+6x_2\le 7 \\ 3x_1+2x_2\le 11 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
11	$f(x)=x_1^2+2x_2^2+x_1+6x_2\to\min$ $\begin{cases} 6x_1+x_2\le 17 \\ x_1+5x_2\le 16 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
12	$f(x)=3x_1^2+2x_2^2+x_1x_2-x_1-2x_2\to\min$ $\begin{cases} 5x_1+x_2\le 17 \\ 4x_1+x_2\le 5 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
13	$f(x)=x_1^2-3x_1+2x_2\to\min$ $\begin{cases} x_1+x_2\le 57 \\ x_1+6x_2\le 26 \end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$

Продовження табл. 6.3

1	2
14	$f(x)=x_2^2-x_1x_2-4x_1-x_2\to\min$ $\begin{cases}4x_1+x_2\le 5\\ x_1+12x_2\le 3\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
15	$f(x)=x_1^2+x_2^2-5x_1x_2-x_1-9x_2\to\min$ $\begin{cases}24x_1+x_2\le 15\\ 3x_1+11x_2\le 23\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
16	$f(x)=x_1^2+5x_2^2+x_1x_2+3x_1+2x_2\to\min$ $\begin{cases}20x_1+12x_2\le 101\\ 43x_1+15x_2\le 120\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
17	$f(x)=x_1^2-x_1x_2+x_1+7x_2\to\min$ $\begin{cases}8x_1+3x_2\le 17\\ x_1+2x_2\le 12\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
18	$f(x)=x_1^2-\frac{1}{2}x_2^2-x_1+6x_2\to\min$ $\begin{cases}x_1+x_2\le 7\\ 4x_1-x_2\le 1\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
19	$f(x)=x_1^2+2x_2^2+x_1x_2-x_1-x_2\to\min$ $\begin{cases}x_1+2x_2\le 7\\ x_1+6x_2\le 15\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
20	$f(x)=x_1^2-5x_1-2x_2\to\min$ $\begin{cases}6x_1+7x_2\le 17\\ 10x_1-x_2\le 66\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$

Закінчення табл. 6.3

1	2
21	$f(x)=x_2^2-x_1x_2-4x_1-x_2\to\min$ $\begin{cases}4x_1+x_2\le 5\\ x_1+12x_2\le 3\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$
22	$f(x)=x_1^2+x_2^2+x_1+2x_2\to\min$ $\begin{cases}3x_1+4x_2\le 13\\ 8x_1+7x_2\le 15\end{cases}$ $x_j\ge 0,\ j=\overline{1,2}$

Функція quadprog надає фінансовим аналітикам, інженерам та дослідникам засоби, необхідні для пошуку оптимальних та компромісних рішень, дозволяє налаштовувати та проводити діагностику задач квадратичної оптимізації і швидко об'єднувати стандартні алгоритми оптимізації за своїми власними методами. Використовуючи результати роботи програм можна зберігати параметри ітераційного процесу і створювати власні критерії зупинки розрахунку. Функція quadprog написана на відкритій мові MATLAB, що дозволяє користувачеві контролювати виконання алгоритму, змінювати вихідний код, а також створювати власні функції і алгоритми.

Лабораторна робота 7
Теорія ігор. Аналіз та управління ризиком в економіці на базі концепції теорії ігор

Мета роботи: набуття практичних навичок щодо побудови математичної моделі матричної гри та розв'язання останньої за допомогою вбудованих функцій середовища MATLAB.

Основні задачі лабораторної роботи

1. Побудувати математичну модель матричної гри.

2. За допомогою вбудованої функції `linprog` середовища MATLAB знайти оптимальний розв'язок задач теорії ігор.

3. За допомогою вбудованої функції `fminimax` середовища MATLAB знайти оптимальний розв'язок економічної задачі, що приводить до теорії ігор.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

7.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

FMINIMAX **Розв'язок задачі мінімакса**

Розв'язати задачу $\min_x \max_F (F_i(x))$ таку, що:

$$c(x) < 0$$
$$ceq(x) = 0$$
$$A \cdot x \le b$$
$$Aeq \cdot x = beq$$
$$lb \le x \le ub,$$

де x, b, beq, lb та ub − вектори;

A і Aeq є матриці;

$c(x)$, $ceq(x)$ і $F(x)$ є такі функції, що повертають вектори.

Синтаксис

```
x = fminimax(fun,x0)
x = fminimax(fun,x0,A,b)
x = fminimax(fun,x0,A,b,Aeq,beq)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
x = fminimax(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon,
options, P1, P2,...)
[x,fval] = fminimax(...)
[x,fval,maxfval] = fminimax(...)
[x,fval,maxfval,exitflag] = fminimax(...)
[x,fval,maxfval,exitflag,output] = fminimax(...)
[x,fval,maxfval,exitflag,output,lambda] = fminimax(...)
```

Опис

`x = fminimax (fun, x0)` починає з точки `x0` і знаходить розв'язок мінімакса від `x` для функції, описаної в `fun`.

`x = fminimax (fun, x0, A, b)` розв'язує задачу мінімакса з умовою лінійних нерівностей `A*x <= b`.

`x = fminimax (fun, x, A, b, Aeq, beq)` розв'язує задачу мінімакса з умовою лінійних рівностей `Aeq*x = beq`. Установлюється `A = []` і `b = []` у разі відсутності нерівностей. `fminimax` мінімізує таким чином, що `c(x) <= 0` і `ceq (x) = 0`. Установлюється `lb = []` або `ub = []` у разі відсутності меж.

`x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)` проводить оптимізацію з урахуванням параметрів оптимізації, визначених у опціях структур.

`x = fminimax (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options, P1, P2, ...)` передає залежні від завдання параметри `P1`, `P2` і т.д., безпосередньо в функції `fun` і `nonlcon`. Передає порожні матриці у вигляді структурних нулів для `A`, `b`, `Aeq`, `beq`, `lb`, `ub`, `nonlcon` і опцій у випадку, якщо ці аргументи не є необхідними.

`[x, fval] = fminimax (...)` повертає значення цільової функції як розв'язок `x`.

`[X, fval, maxfval] = fminimax (...)` повертає максимальне значення функції як розв'язок `x`.

`[X, fval, maxfval, exitflag] = fminimax (...)` повертає значення `exitflag`, яке містить вихідні умови для `fminimax`.

`[X, fval, maxfval, exitflag, output] = fminimax (...)` повертає структурний вихід з інформацією про оптимізацію.

`[X, fval, maxfval, exitflag, output, lambda] = fminimax (...)` повертає структурну `lambda`, поля якої містять множники Лагранжа, як розв'язок `x`.

Аргументи

Табл. 7.1 містить загальний опис переданих в `fminimax` аргументів.

Таблиця 7.1

Вхідні аргументи функції `fminimax`

Аргументи	Опис
`fun`	`fun` є якась функція, яка приймає вектор `x` і повертає вектор `F`, як цільову функцію від `x`
`nonlcon`	Функція `nonlcon` приймає вектор `x` і повертає два вектора `c` і `ceq`. Вектор `c` містить нелінійні нерівності у розрахунку від `x`, і `ceq` містить нелінійні рівності у розрахунку від `x`. Функція `nonlcon` може бути визначена як описувач функції
`options`	Опції забезпечують облік специфічних деталей функції у вигляді параметрів `options`

У табл. 7.2 наведено загальний опис вихідних параметрів `fminimax`, які мають ряд особливостей.

Таблиця 7.2

Вихідні аргументи функції `fminimax`

Параметри	Опис	
`exitflag`	Певне ціле число, що ідентифікує причину зупинення алгоритму. Перелік прийнятих значень і відповідних причин зупинки алгоритму:	
	> 0	функція зійшлася до деякого розв'язку `x`
	0	максимальне число оцінки функції або ітерацій було перевищено
	< 0	ця задача не має розв'язку
`lamda`	Структура, яка містить множники Лагранжа у розв'язанні за `x` (розподіленому за типами станів). Поле структури:	
	`Lower`	нижні межі `lb`
	`Upper`	верхні межі `ub`
	`Ineqlin`	лінійні нерівності
	`Eqlin`	лінійні рівності
`output`	Структура, яка містить інформацію про оптимізацію. Поле структури:	
	`iterations`	число виконаних ітерацій
	`funcCount`	число оцінок функції
	`algorithm`	використовуваний алгоритм

Опції

Параметри оптимізаційних опцій, які використовуються в `fminimax` приведені у табл. 7.3. Можна використовувати `optimset` для того, щоб установити або змінити значення даних полів в структурі параметрів опцій.

Таблиця 7.3

Параметри оптимізаційних опцій

Назва параметру оптимізації	Опис
DerivativeCheck	Порівняння похідних, що вводяться користувачем (градієнти мети або обмежуючих умов) з кінцево-різницевими похідними
Diagnostics	Друкування діагностичної інформації про функції, що підлягає мінімізації або розв'язанню
DiffMaxChange	Максимальна зміна в змінних для кінцево-різницевих градієнтів
DiffMinChange	Мінімальна зміна в змінних для кінцево-різницевих градієнтів
Display	Рівень відображення. 'Off' відображення не здійснюється; 'iter' відображення проводиться на кожній ітерації; 'final' (приймається за замовчуванням) відображення тільки кінцевої інформації
GradConstr	Градієнт для певних обмежень
GradObj	Градієнт для визначених користувачем обмежень цільової функції. Для того, що б використовувати крупно-масштабний метод необхідно ставити зазначений градієнт. Це необов'язково для середньомасштабного методу
MaxFunEvals	Максимальне число допустимих розрахунків функції.
MaxIter	Максимальне число допустимих ітерацій
MeritFunction	Для установлення 'multiobj' використовується цільова (goal) функція корисності досягнення / мінімакс. Для установлення 'singleobj' використовується функція корисності fmincon
MinAbsMax	Число функцій F(x), що підлягають мінімізації в найгіршому випадку для абсолютних значень
TolCon	Кінцеве допустиме відхилення щодо порушення умов обмеження
TolFun	Кінцеве допустиме відхилення за значенням функції
TolX	Кінцеве допустиме відхилення за значенням x

Для більшої деталізації використовуваного алгоритму і типів процедур перейдемо до розв'язання практичних задач.

7.2. Розв'язування типових задач у середовищі MATLAB

Приклад 7.1. Задана платіжна матриця P, елементи якої визначають виграш продавця (гравець A) під час реалізації однієї з його можливих стратегій за умови, що покупець (гравець B) дотримується однієї із своїх можливих стратегій:

$$P = \begin{pmatrix} 5 & 6 & 2 \\ 7 & 3 & 7 \\ 6 & 4 & 5 \\ 9 & 12 & 4 \\ 8 & 7 & 6 \end{pmatrix}.$$

Знайдіть ймовірності, за якими покупець A і продавець B дотримуються своїх стратегій (оптимальні плани обох гравців) для досягнення оптимальної вартості покупки, а також вартість покупки (ціну гри) за допомогою програмного середовища *MATLAB*.

Розв'язання. Побудуємо математичну модель задачі, розглядаючи її з позиції гравця A (пряма задача). Оскільки платіжна матриця має розмір 5×3, то гравець A має п'ять стратегій, яким поставимо у відповідність матрицю-стовпець ймовірностей $P=(p_1, p_2, p_3, p_4, p_5)^T$. Це змінні прямої задачі. Гравець B (двоїста задача) має три стратегії, які описуються матрицею-стовпцем ймовірностей $Q=(q_1, q_2, q_3)^T$.

Для прямої задачі система обмежень має вигляд:

$$\begin{cases} 5p_1+7p_2+6p_3+9p_4+8p_5 \ge \nu; \\ 6p_1+3p_2+4p_3+12p_4+7p_5 \ge \nu; \\ 2p_1+7p_2+5p_3+4p_4+6p_5 \ge \nu; \\ p_1+p_2+p_3+p_4+p_5=1; \\ 0 \le p_i \le 1; \quad i=\overline{1,5}. \end{cases}$$

Введемо нові змінні $x_i = p_i / \nu$ $(i=\overline{1,5})$ і за їхньою допомогою запишемо систему обмежень допоміжної задачі, яка є задачею лінійного програмування:

$$\begin{cases} 5x_1+7x_2+6x_3+9x_4+8x_5 \ge 1; \\ 6x_1+3x_2+4x_3+12x_4+7x_5 \ge 1; \\ 2x_1+7x_2+5x_3+4x_4+6x_5 \ge 1; \\ x_i \ge 0; \quad i=\overline{1,5}. \end{cases}$$

Цільова функція цієї задачі має вигляд:

$$z(x)=x_1+x_2+x_3+x_4+x_5 \to \min.$$

Для двоїстої задачі система обмежень має вигляд:

$$\begin{cases} 5y_1 + 6y_2 + 2y_3 \le 1; \\ 7y_1 + 3y_2 + 7y_3 \le 1; \\ 6y_1 + 4y_2 + 5y_3 \le 1; \\ 9y_1 + 12y_2 + 4y_3 \le 1; \\ 8y_1 + 7y_2 + 6y_3 \le 1; \end{cases}$$
$$y_j \ge 0; \quad j = \overline{1,3}.$$

Цільова функція цієї задачі має вигляд:

$$g(y) = y_1 + y_2 + y_3 \to \max.$$

Вирішимо в MATLAB задачу лінійного програмування. Відповідна програма (m-файл) виглядає так:

```
clc
clear all
%Платіжна матриця для гравця A
A =[5 7 6 9 8; 6 3 4 12 7; 2 7 5 4 6];
%Платіжна матриця для гравця B
B = A';
%Розв'язання матричної гри:
[x,fval,exitflag,output,lambda] = linprog(ones(5,1), -A, -
ones(3,1), [], [],zeros(5,1));
[y,gval,exitflag,output,lambda] = linprog(-ones(3,1), B,
ones(5,1), [], [],zeros(3,1));
v=1/fval; % Ціна гри
rats(v)
p=v*x; % Оптимальна стратегія гравця A
rats(p)
q=v*y; % Оптимальна стратегія гравця B
rats(q)
```

Запустивши програму, отримаємо:

```
Optimization terminated.

v = 31/5
```

```
P =
         0
       1/5
         0
         0
       4/5

Q =
         0
       1/5
       4/5
```

Таким чином, оптимальна змішана стратегія гравця A: $P=\left(0;\frac{1}{5};0;0;\frac{4}{5}\right)$; оптимальна змішана стратегія гравця B: $Q=\left(0;\frac{1}{5};\frac{4}{5}\right)$; ціна гри: $v=\frac{31}{5}$.

Приклад 7.2. Швейне підприємство (надалі – фірма) реалізує свою продукцію через магазин. Збут залежить від стану погоди. В умовах теплої погоди підприємство реалізує 1 000 костюмів і 2 300 суконь, а за прохолодної − 1 400 костюмів і 700 суконь. Витрати на виготовлення одного костюма дорівнюють 20 ум. од., а сукні − 5 ум. од.; ціна реалізації, відповідно, дорівнює 40 і 12 ум. од. Визначте оптимальну стратегію підприємства.

Розв'язання. Побудуємо математичну модель задачі. У зв'язку з можливими станами попиту фірма має у своєму розпорядженні дві стратегії:

$F_1=(1\,000,\,2\,300)$ − виробити 1 000 костюмів і 2 300 суконь;

$F_2=(1\,400,\,700)$ − виробити 1 400 костюмів і 700 суконь.

Природа (ринок) має також дві стратегії:

D_1 = Погода тепла; D_2 = Погода прохолодна.

Якщо фірма прийме стратегію F_1 і попит дійсно буде перебувати в першому стані, тобто погода буде теплою (D_1), то випущена продукція буде повністю реалізована, дохід буде складати: $p_{11}=1\,000\cdot(40-20)+2\,300\cdot(12-5)=36\,100$.

Якщо фірма прийме стратегію F_1, а попит буде перебувати в стані D_2 (погода прохолодна), то сукні будуть реалізовані лише частково, і дохід складе: $p_{12}=1\ 000\cdot(40-20)+700\cdot(12-5)-(2\,300-700)\cdot 5=16\,900$.

Аналогічно, якщо фірма вибере стратегію F_2, а природа − стратегію D_1 (погода тепла), то дохід складе (будуть недорозпродані костюми):

$p_{21} = 1\,000 \cdot (40 - 20) + 700 \cdot (12 - 5) - (1\,400 - 1\,000) \cdot 20 = 16\,900$, а якщо природа вибере стратегію D_2, то $p_{22} = 1\,400 \cdot (40 - 20) + 700 \cdot (12 - 5) = 32\,900$.

Розглядаючи фірму та природу як двох гравців, отримаємо платіжну матрицю гри:

$$P = \begin{pmatrix} 36\,100 & 16\,900 \\ 16\,900 & 32\,900 \end{pmatrix}.$$

Математичні моделі пари двоїстих задач лінійного програмування можна записати так:

знайти мінімум функції $F(x)$: $F(x) = x_1 + x_2 \to \min$ з обмеженнями (для другого гравця):

$$\begin{cases} 36\,100x_1 + 16\,900x_2 \geq 1 \\ 16\,900x_1 + 32\,900x_2 \geq 1 \end{cases};$$

знайти максимум функції $Z(y)$: $F(y) = y_1 + y_2 \to \max$ з обмеженнями (для першого гравця):

$$\begin{cases} 36\,100y_1 + 16\,900y_2 \leq 1 \\ 16\,900y_1 + 32\,900y_2 \leq 1 \end{cases}.$$

Розв'яжемо пару двоїстих задач у програмному середовищі MATLAB. Відповідна програма (m-файл) виглядає так:

```
clc
clear all
%Платіжна матриця для гравця A
A =[36100 16900; 16900 32900];
%Платіжна матриця для гравця B
B = A';
%Розв'язання матричної гри:
[x,fval,exitflag,output,lambda]    =    linprog(ones(2,1),    -A,    -
ones(2,1), [], [], zeros(2,1))
[y,gval,exitflag,output,lambda]     =     linprog(-ones(2,1),     B,
ones(2,1), [], [], zeros(2,1))
v=1/fval; % Ціна гри
rats(v)
p=v*x; % Оптимальна стратегія гравця A
rats(p)
```

```
q=v*y; % Оптимальна стратегія гравця B
rats(q)
```

Запустивши програму, отримаємо

```
Optimization terminated.
exitflag = 1
output =

      iterations: 7
       algorithm: 'large-scale: interior point'
    cgiterations: 0
         message: 'Optimization terminated.'

v = 281900/11

P =
      5/11
      6/11

Q =
      5/11
      6/11
```

Оптимальна стратегія фірми: $F = x_1F_1 + x_2F_2 = \frac{5}{11}(1\,000, 2\,300) + \frac{6}{11}(1\,400, 700) = \left(\frac{13\,400}{11}, \frac{15\,700}{11}\right) = (1\,218, 1\,427)$. Таким чином, фірмі необхідно виробити 1 218 костюмів і 1 427 суконь.

Запитання для самоперевірки

1. Що таке матрична гра? Поясніть її економічну змістовність.
2. До якого класу задач математичного програмування належать задачі, які базуються на матричної грі?
3. Яка функція програмного середовища MATLAB застосовується до розв'язання матричних ігор?
4. Опишіть процес побудови математичної моделі задачі теорії ігор.
5. Яке призначення функції `fminimax`?

6. Назвіть вхідні параметри функції `fminimax`.

7. Назвіть вихідні параметри функції `fminimax`.

8. Назвіть оптимізаційні задачі, що зводяться до матричних ігор.

Завдання до лабораторної роботи

Для розв'язання задач теорії матричних ігор необхідно:

1) побудувати математичну модель задачі теорії ігор, заданої платіжною матрицею;

2) за допомогою вбудованої функції `linprog` середовища MATLAB знайти оптимальний розв'язок прямої та двоїстої задачі теорії ігор;

3) змінити алгоритм вбудованої функції `linprog` за допомогою функції `optimset`. Провести порівняльний аналіз отриманих результатів. Зробити висновки;

4) провести економічний аналіз отриманих результатів.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Варіанти завдань наведені у табл. 7.4.

Таблиця 7.4

Варіанти завдань до лабораторної роботи 7

№ Варіанта	Платіжна матриця
1	2
1	$P=\begin{pmatrix} 4 & 5 & 11 & 2 & 3 \\ 6 & 8 & 10 & 6 & 5 \\ 4 & 7 & 9 & 8 & 4 \\ 3 & 8 & 11 & 9 & 6 \\ 5 & 2 & 8 & 7 & 2 \end{pmatrix}$
2	$P=\begin{pmatrix} 9 & 5 & 6 & 9 & 3 \\ 6 & 8 & 8 & 6 & 5 \\ 11 & 7 & 9 & 9 & 7 \\ 3 & 8 & 5 & 9 & 6 \\ 12 & 2 & 8 & 7 & 8 \end{pmatrix}$

Продовження табл. 7.4

1	2
3	$P = \begin{pmatrix} 9 & 15 & 6 & 9 & 3 \\ 7 & 10 & 8 & 6 & 5 \\ 4 & 10 & 9 & 9 & 7 \\ 3 & 8 & 5 & 2 & 6 \\ 2 & 12 & 8 & 7 & 3 \end{pmatrix}$
4	$P = \begin{pmatrix} 12 & 5 & 6 & 9 & 6 \\ 3 & 4 & 8 & 14 & 5 \\ 4 & 6 & 9 & 11 & 3 \\ 5 & 8 & 5 & 2 & 6 \\ 10 & 9 & 8 & 9 & 7 \end{pmatrix}$
5	$P = \begin{pmatrix} 5 & 8 & 6 & 12 & 7 \\ 6 & 12 & 5 & 7 & 8 \\ 11 & 7 & 9 & 10 & 7 \\ 3 & 14 & 5 & 9 & 4 \\ 12 & 2 & 3 & 7 & 9 \end{pmatrix}$
6	$P = \begin{pmatrix} 9 & 5 & 6 & 9 & 3 \\ 10 & 8 & 10 & 6 & 9 \\ 8 & 15 & 9 & 10 & 7 \\ 6 & 12 & 5 & 13 & 8 \\ 12 & 9 & 11 & 7 & 6 \end{pmatrix}$
7	$P = \begin{pmatrix} 11 & 5 & 7 & 9 & 6 \\ 3 & 4 & 9 & 14 & 5 \\ 12 & 6 & 8 & 11 & 8 \\ 8 & 11 & 5 & 10 & 6 \\ 6 & 10 & 8 & 9 & 5 \end{pmatrix}$
8	$P = \begin{pmatrix} 12 & 5 & 6 & 9 & 6 \\ 11 & 6 & 8 & 14 & 5 \\ 4 & 7 & 9 & 11 & 8 \\ 5 & 8 & 5 & 2 & 6 \\ 10 & 9 & 8 & 9 & 4 \end{pmatrix}$

Продовження табл. 7.4

1	2
9	$P=\begin{pmatrix}11 & 5 & 6 & 9 & 6\\ 8 & 4 & 7 & 8 & 10\\ 9 & 6 & 9 & 7 & 13\\ 12 & 8 & 5 & 2 & 6\\ 8 & 9 & 8 & 6 & 10\end{pmatrix}$
10	$P=\begin{pmatrix}10 & 5 & 6 & 9 & 8\\ 6 & 8 & 11 & 6 & 5\\ 11 & 7 & 9 & 8 & 7\\ 9 & 8 & 10 & 9 & 6\\ 12 & 4 & 8 & 7 & 9\end{pmatrix}$
11	$P=\begin{pmatrix}8 & 5 & 10 & 9 & 6\\ 13 & 4 & 8 & 8 & 5\\ 9 & 6 & 11 & 6 & 8\\ 5 & 8 & 5 & 2 & 6\\ 10 & 9 & 8 & 6 & 9\end{pmatrix}$
12	$P=\begin{pmatrix}12 & 5 & 13 & 9 & 6\\ 11 & 4 & 8 & 7 & 5\\ 4 & 6 & 12 & 8 & 9\\ 10 & 8 & 5 & 2 & 6\\ 7 & 9 & 8 & 6 & 7\end{pmatrix}$
13	$P=\begin{pmatrix}12 & 5 & 6 & 9 & 7\\ 9 & 4 & 8 & 10 & 5\\ 11 & 6 & 9 & 11 & 9\\ 5 & 8 & 7 & 12 & 8\\ 10 & 9 & 8 & 9 & 6\end{pmatrix}$
14	$P=\begin{pmatrix}9 & 11 & 6 & 9 & 8\\ 8 & 9 & 11 & 6 & 7\\ 7 & 12 & 9 & 10 & 5\\ 6 & 11 & 5 & 13 & 8\\ 5 & 9 & 13 & 7 & 4\end{pmatrix}$

Продовження табл. 7.4

1	2
15	$P=\begin{pmatrix} 8 & 5 & 6 & 9 & 6 \\ 3 & 4 & 8 & 10 & 5 \\ 4 & 6 & 9 & 5 & 7 \\ 12 & 8 & 5 & 14 & 6 \\ 11 & 9 & 8 & 9 & 4 \end{pmatrix}$
16	$P=\begin{pmatrix} 9 & 8 & 6 & 7 & 6 \\ 12 & 8 & 10 & 6 & 9 \\ 8 & 15 & 9 & 10 & 8 \\ 10 & 8 & 12 & 13 & 4 \\ 14 & 7 & 8 & 11 & 6 \end{pmatrix}$
17	$P=\begin{pmatrix} 9 & 5 & 6 & 14 & 6 \\ 3 & 8 & 11 & 6 & 0 \\ 8 & 15 & 9 & 10 & 2 \\ 10 & 9 & 12 & 13 & 4 \\ 4 & 7 & 8 & 1 & 6 \end{pmatrix}$
18	$P=\begin{pmatrix} 5 & 5 & 6 & 5 & 6 \\ 2 & 18 & 11 & 6 & 10 \\ 9 & 17 & 9 & 0 & 2 \\ 6 & 9 & 12 & 4 & 7 \\ 3 & 7 & 20 & 1 & 6 \end{pmatrix}$
19	$P=\begin{pmatrix} 10 & 5 & 6 & 9 & 8 \\ 6 & 8 & 2 & 6 & 5 \\ 11 & 7 & 9 & 8 & 7 \\ 9 & 8 & 1 & 9 & 6 \\ 3 & 4 & 8 & 7 & 19 \end{pmatrix}$
20	$P=\begin{pmatrix} 1 & 4 & 6 & 7 & 3 \\ 6 & 18 & 3 & 6 & 5 \\ 2 & 7 & 9 & 8 & 7 \\ 9 & 8 & 10 & 9 & 16 \\ 12 & 4 & 8 & 7 & 9 \end{pmatrix}$

Закінчення табл. 7.4

1	2
21	$P = \begin{pmatrix} 6 & 15 & 6 & 9 & 13 \\ 7 & 10 & 8 & 6 & 5 \\ 4 & 21 & 7 & 9 & 7 \\ 13 & 8 & 15 & 5 & 9 \\ 2 & 10 & 8 & 7 & 3 \end{pmatrix}$

Представлено метод реалізації оптимальних змішаних стратегій у матричній грі, котрий полягає у використанні вбудованих функцій `linprog` і `fminimax`.

Лабораторна робота 8
Динамічне програмування

Мета роботи: набуття практичних навичок щодо розв'язання задач динамічного програмування, а саме – задачі управління запасами та задач, які розв'язуються на основі ідей динамічного програмування, за допомогою програмного середовища MATLAB.

Основні задачі лабораторної роботи

1. Розглянути основні задачі динамічного програмування.

2. Для розв'язання задачі розподілу інвестицій скласти програму у середовищі MATLAB.

3. Щоб переконатися, що програма правильно розв'язує задачі методом динамічного програмування, написати також програми повного перебору − для перевірки основного алгоритму та для більш ясного розуміння кожної з розглянутих економічних задач.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

8.1. Теоретичні відомості про функції MATLAB, використовувані в лабораторній роботі

Лабораторна робота присвячена розв'язанню та програмній реалізації одного з видів задач теорії управління, а саме – задачі розподілу інвестицій методом динамічного програмування.

Рівняння Беллмана (також відоме як рівняння динамічного програмування), назване на честь Річарда Ернста Беллмана, є достатньою умовою для оптимальності, що асоціюється з математичним методом оптимізації – динамічним програмуванням, яке базується на *Принципі оптимальності Беллмана*. Рівняння Беллмана є диференціальним рівнянням у частинних похідних з початковими умовами, заданими для останнього моменту часу (тобто праворуч), для функції Беллмана, яка виражає мінімальне значення критерію оптимізації, яке може бути досягнуте за умови еволюції системи з поточного стану в деякий кінцевий. Це, в свою чергу, дозволяє перейти від розв'язання вихідної багатокрокової задачі оптимізації до послідовного розв'язування кількох однокрокових оптимізаційних задач.

Поняття рівняння Беллмана та функції Беллмана застосовується тільки для неперервних систем. Для дискретних систем аналогом виступає так зване *основне рекурентне співвідношення*, що є формальною основою методу динамічного програмування і виражає достатню умову оптимальності, та функція майбутніх втрат.

На жаль, вбудована функція розв'язання задач динамічного програмування за допомогою функції Беллмана в програмному середовищі MATLAB відсутня. Тому у посібнику приводиться готова програмна реалізація цієї функції як окремий m-файл (Bellman.m), яким зручно користуватися для розв'язання власних задач розподілу інвестицій.

```
format compact; clc % очистка екрану
y1; % кількість інвестицій
C = [] % витрати
R = [] % майбутній дохід
[u9, n] = size(C); % кількість можливих управлінь
x_min = ones(1, n)
x_max = sum(finite(R))
x = x_min; x_opt = x_min;
f_opt = -inf;
```

```
j = n;
while 1
    if x(j) <= x_max(j)
        cost = 0;
        for i = 1:n
            cost = cost + C(x(i), i);
        end
        if cost <= y1
            f = 0;
            for i = 1:n
                f = f + R(x(i), i);
            end
            if f_opt < f
                f_opt = f
                x_opt = x
            elseif f_opt == f
                    f_opt
                    x_opt = [x_opt; x]
            end
        end % cost
        j = n;
    else
        x(j) = x_min(j);
        j = j - 1;
        if j <= 1e-5
            break
        end
    end % if
   x(j) = x(j) + 1;
end % while 1
disp('ANSWER :')
f_opt, x_opt
```

8.2. Розв'язування типових задач у середовищі MATLAB

Приклад 8.1 (задача розподілу інвестицій). Рада директорів фірми вивчає пропозиції щодо модернізації чотирьох підприємств. Для цих цілей виділено 8 мільйонів доларів. Для кожного підприємства j розроблено кілька альтернативних проектів. Кожен з проектів характеризується сумарними витратами c_j і майбутніми доходами R_j. На кожному підприємстві можна реалізувати тільки один проект. Відповідні дані наведені в табл. 8.1.

Таблиця 8.1

Задача розподілу восьми мільйонів доларів

Проект	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	1	3	3	5	1	4	2	3
$x_j = 3$	–	–	5	9	2	6	–	–

Необхідно вибрати такі проекти для кожного підприємства, щоб фірма отримала максимальний річний дохід.

Розв'язання. Складемо математичну модель задачі.

$$\begin{cases} f_1(y_1) = \max\limits_{x_1,\dots,x_n} \left\{ \sum\limits_{j=1}^{n} R_j(x_j) \right\} \\ \sum\limits_{j=1}^{n} c_j(x_j) \le y_1 \\ x_j - \textit{цілі},\ j = \overline{1,n}, \end{cases} \tag{8.1}$$

де y_j – кількість грошей, виділених для розширення підприємств $j, j+1, \dots, n$;

x_j – номер проекту, обраного на підприємстві j;

$c_j(x_j)$ – витрати на $j-му$ підприємстві для проекту x_j. Вимірюється в мільйонах доларів;

$R_j(x_j)$ – річний дохід, який буде отриманий від реалізації проекту x_j. Вимірюється в мільйонах доларів на рік;

$f_1(y_1)$ – максимальний річний дохід, який буде отриманий від реалізації проектів $x_1, x_2, \dots, x_n$ із заданим обсягом інвестицій y_1 мільйонів доларів.

У задачі (8.1) оберемо спочатку $x_1 = \tilde{x}_1 = 1$:

$$\begin{cases} f_1(y_1) = R_1(\tilde{x}_1) + \max\limits_{x_2,\dots,x_n} \left\{ \sum\limits_{j=2}^{n} R_j(x_j) \right\} \\ \sum\limits_{j=2}^{n} c_j(x_j) \le y_1 \\ x_j - \textit{цілі},\ j = \overline{2,n} \end{cases} \quad \text{або} \quad \tilde{f}_1(y_1) = R_1(\tilde{x}_1) + f_2(y_1 - c_1(\tilde{x}_1)),$$

де $\tilde{f}_1(y_1)$ – найбільший річний дохід з інвестиціями y_1 і фіксованим значенням $\tilde{x}_1 = 1$;

$f_2(y_1 - c_1(\tilde{x}_1))$ – річний дохід від розширення підприємств $2,...,n$, якщо інвестовано грошей $y_1 - c_1(\tilde{x}_1)$ мільйонів доларів.

Далі нехай $x_1 = \tilde{x}_1 = 2$. У результаті отримаємо повний перебір, отже отримаємо оптимальне значення $f_1(y_1)$:

$$f_1(y_1) = \max_{x_1 | c_1(x_1) \le y_1} \{R_1(x_1) + f_2(y_1 - c_1(x_1))\}. \qquad (8.2)$$

Легко помітити, що отримана формула (8.2) дозволяє розв'язувати задачу розподілу інвестицій для $n-1$ підприємства $(2,...,n)$ щодо всіх можливих значень інвестицій $y_1 = 0,1,...,5$. Що залишилося, слід записати у формулу (8.2) у загальному випадку, для обчислення максимального прибутку від модернізації підприємств $j, j+1,...,n$:

$$\begin{cases} f_{n+1}(y_{n+1}) = 0, \\ f_j(y_j) = \max\limits_{x_j | c_j(x_j) \le y_j} \{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\}. \\ j = n, n-1,...,1 \end{cases} \qquad (8.3)$$

Залежність (8.3) є рівнянням Беллмана для процедури зворотного прогону.

Змінні в динамічному програмуванні мають свої назви:

x_j – змінна (або управління) Беллмана;

y_j – параметр (стан у термінології Беллмана) – кількість грошей для підприємств $j,...,n$ у даній задачі про розподіл інвестицій.

Рекурентне співвідношення Беллмана для процедури зворотного прогону записується таким чином для задачі 8.1 (див. формулу 8.3):

$$f_8(y_8) = 0,$$
$$f_j(y_j) = \max_{x_j | c_j(x_j) \le y_j} \{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\}, \; j = \overline{4,1}.$$

Дії, які необхідно виконати для розв'язування поставленої задачі за допомогою функції Беллмана, наведені у відповідній програмі (m-файл), яка виглядає так:

```
format compact; clc
y1 = 8;
```

```
C = [1 2 1 1
     3 4 2 3
  -inf -inf 4 -inf]
R = [1 2 1 3
     4 6 3 5
  -inf -inf 7 -inf]
[u9, n] = size(C);
x_min = ones(1, n)
x_max = sum(finite(R))
x = x_min; x_opt = x_min;
f_opt = -inf;
j = n;
while 1
    if x(j) <= x_max(j)
        cost = 0;
        for i = 1:n
            cost = cost + C(x(i), i);
        end
        if cost <= y1
            f = 0;
            for i = 1:n
                f = f + R(x(i), i);
            end
            if f_opt < f
                f_opt = f
                x_opt = x
            elseif f_opt == f
                    f_opt
                    x_opt = [x_opt; x]
            end
        end % cost
        j = n;
    else
        x(j) = x_min(j);
        j = j - 1;
        if j <= 1e-5
            break
        end
    end % if
   x(j) = x(j) + 1;
end % while 1
disp('ANSWER :')
f_opt, x_opt
```

Запустивши програму, отримаємо:

```
ANSWER :
f_opt =
    13
x_opt =
     1     1     3     1
     1     2     2     1
```

Таким чином, щоб оптимальним чином витратити 8 мільйонів доларів на чотирьох підприємствах, необхідно обрати проекти:

$$x^* = (1, 1, 3, 1) \text{ або } x^* = (1, 2, 2, 1).$$

$f_1(8) = 13$ мільйонів доларів у рік складе максимальний щорічний прибуток.

Запитання для самоперевірки

1. Що таке динамічне програмування? Чи завжди можна застосувати метод динамічного програмування?

2. Які задачі розв'язуються за допомогою динамічного програмування? Наведіть приклади.

3. Сформулюйте принцип оптимальності Беллмана.

4. Яка функція програмного середовища MATLAB застосовується до розв'язання задач динамічного програмування? Опишіть алгоритм.

5. Поясніть процес побудови математичної моделі задачі розподілу інвестицій.

Завдання до лабораторної роботи

Для розв'язання задач розподілу інвестицій необхідно:

1) побудувати математичну модель задачі. Покласти y_1 дорівненим номеру варіанта студента;

2) записати рекурентне співвідношення Беллмана для процедури зворотного прогону заданої задачі;

3) за допомогою функції `Bellman.m` середовища MATLAB знайти оптимальний розв'язок задачі розподілу інвестицій;

4) замінити `Bellman.m` програмою повного перебору для перевірки основного алгоритму та для більш ясного розуміння кожної з розглянутих економічних задач. Зробити висновки;

5) провести економічний аналіз отриманих результатів.

Кожна лабораторна робота повинна бути окремим робочим модулем, написаним у m-файлі.

Варіанти завдань наведені у табл. 8.2.

Таблиця 8.2

Варіанти завдань до лабораторної роботи 8

Варіант 1								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	5	1	6	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 2								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	5	4	7	2	3	6	2	3
$x_j = 3$	–	–	11	4	4	8	–	–
Варіант 3								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	5	8	6	2	3
$x_j = 3$	–	–	2	3	4	2	–	–
Варіант 4								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	6	5	1	5	1	6	2	13
$x_j = 3$	–	–	9	1	4	2	–	–

Варіант 5								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	1	6	3	5	1	6	2	3
$x_j = 3$	–	–	7	4	1	2	–	–
Варіант 6								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	5	1	7	2	1
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 7								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	2	8	5	3	6	4	3
$x_j = 3$	–	–	9	1	1	1	–	–
Варіант 8								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	8	2	4	5	1	6	2	3
$x_j = 3$	–	–	9	1	5	2	–	–
Варіант 9								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	5	5	3	5	4	6	2	3
$x_j = 3$	–	–	9	1	2	2	–	–

Варіант 10								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	1	6	5	1	6	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 11								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	5	1	6	2	3
$x_j = 3$	–	–	9	1	41	2	–	–
Варіант 12								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	7	3	5	1	5	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 13								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	1	5	4	6	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 14								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	6	1	2	2	11
$x_j = 3$	–	–	9	1	1	2	–	–

Варіант 15								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	5	1	6	2	3
$x_j = 3$	–	–	10	1	2	1	–	–
Варіант 16								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	3	6	5	8	2	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 17								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	1	5	6	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 18								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	9	5	4	6	2	3
$x_j = 3$	–	–	9	1	1	2	–	–
Варіант 19								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	4	5	3	2	1	6	2	3
$x_j = 3$	–	–	7	1	1	2	–	–

Закінчення табл. 8.2

Варіант 20								
Проекти	Перше підприємство		Друге підприємство		Третє підприємство		Четверте підприємство	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	3	5	3	6	2	9	1	1
$x_j = 3$	–	–	8	1	1	2	–	–

Метод динамічного програмування полягає в тому, що оптимальне управління будується поступово. На кожному кроці оптимізується управління тільки цього кроку. На кожному кроці управління вибирається з урахуванням наслідків, так як управління, що оптимізує цільову функцію тільки для даного кроку, може привести до неоптимальному ефекту усього процесу. Управління на кожному кроці має бути оптимальним з точки зору процесу в цілому.

Який би не був початковий стан системи перед черговим кроком, управління на цьому етапі вибирається так, щоб виграш на даному кроці плюс оптимальний виграш на всіх наступних кроках був максимальним. Це наочно відображено в Bellman.m.

Використана література

1. Малярець Л. М. Економіко-математичні методи та моделі : навч. посіб. / Л. М. Малярець. – Харків : Вид ХНЕУ ім. С. Кузнеця, 2014. – 412 с.

2. Малярець Л. М. Економіко-математичні методи і моделі : навч.-практ. посіб. / Л. М. Малярець, Е. Ю. Железнякова, Є. Ю. Місюра. – Харків : Вид. ХНЕУ, 2011. – 320 с.

3. Малярец Л. М. Экономико-математические методы и модели : учеб. пособ. для иностранных студентов / Л. М. Малярец. – Харьков : Изд. ХНЭУ, 2013. – 288 с.

4. Малярець Л. М. Сучасні оптимізаційні методи в середовищі MatLab : навч. посіб. у 2-х ч. / Л. М. Малярець, Є. В. Рєзнік, Б. В. Сінкевич. – Харків : Вид. ХНЕУ, 2011. – Ч.1. – 360 с.

5. Малярець Л. М. Сучасні оптимізаційні методи в середовищі MatLab : навч. посіб. у 2-х ч. / Л. М. Малярець, Є. В. Рєзнік, Б. В. Сінкевич. – Харків : Вид. ХНЕУ, 2013. – Ч.2. – 356 с.

6. Wetherbe J. C. (1979). Systems analysis for computer-based information systems, West series in data processing and information systems, West Pub. Co., ISBN 9780829902280.

7. Michael R. Garey and David S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman. ISBN 0-7167-1045-5. A2.5: ND43, pg.218.

8. Mortensen, D. (1986). "Job search and labor market analysis". In Ashenfelter, O.; Card, D. The Handbook of Labor Economics. 2. Amsterdam: North-Holland. ISBN 0-444-87857-2.

9. Lucas, R.; Stokey, N. (1989). Recursive Methods in Economic Dynamics. Cambridge: Harvard University Press. pp. 304–315. ISBN 0-674-75096-9.

10. Adda, J.; Cooper, R. (2003). Dynamic Economics: Quantitative Methods and Applications. Cambridge: MIT Press. p. 257. ISBN 0-262-01201-4.

Додатки

Додаток А
Таблиця А.1

Параметри оптимізації

Ім'я параметра	Опис	Використовується у функціях	Обмеження
1	2	3	4
AbsoluteGapTolerance	Nonnegative real. intlinprog stops if the difference between the internally calculated upper (U) and lower (L) bounds on the objective function is less than or equal to AbsoluteGapTolerance: U – L <= AbsoluteGapTolerance.	intlinprog	optimoptions only
AbsoluteMaxObjectiveCount	Number of F(x) to minimize the worst case absolute values.	fminimax	
Algorithm	Chooses the algorithm used by the solver.	fmincon, fminunc, fsolve, linprog, lsqcurvefit, lsqlin, lsqnonlin, quadprog	
BranchRule	Rule for choosing the component for branching: 'maxpscost' — The fractional component with maximum pseudocost. See Branch and Bound. 'mostfractional' — The component whose fractional part is closest to 1/2. 'maxfun' — The fractional component with maximal corresponding component in the absolute value of objective vector f.	intlinprog	optimoptions only
CheckGradients	Compare user-supplied analytic derivatives (gradients or Jacobian, depending on the selected solver) to finite differencing derivatives.	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin	optimoptions only. For optimset, use DerivativeCheck

Продовження табл. А.1

1	2	3	4
ConstraintTolerance	Tolerance on the constraint violation.	fgoalattain, fmincon, fminimax, fseminf, intlinprog, linprog, lsqlin, quadprog	optimoptions only. For optimset, use TolCon
CutGeneration	Level of cut generation (see Cut Generation): 'none' — No cuts. Makes CutMaxIterations irrelevant. 'basic' — Normal cut generation. 'intermediate' — Use more cut types. 'advanced' — Use most cut types.	intlinprog	optimoptions only
CutMaxIterations	Number of passes through all cut generation methods before entering the branch-and-bound phase, an integer from 1 through 50. Disable cut generation by setting the CutGeneration option to 'none'.	intlinprog	optimoptions only
Display	Level of display. 'off' displays no output. 'iter' displays output at each iteration, and gives the default exit message. 'iter-detailed' displays output at each iteration, and gives the technical exit message. 'notify' displays output only if the function does not converge, and gives the default exit message. 'notify-detailed' displays output only if the function does not converge, and gives the technical exit message. 'final' displays just the final output, and gives the default exit message. 'final-detailed' displays just the final output, and gives the technical exit message.	All. See the individual function reference pages for the values that apply.	

Продовження табл. А.1

1	2	3	4
EqualityGoalCount	Specify the number of objectives required for the objective fun to equal the set goal. Reorder your objectives, if necessary, to have fgoalattain achieve the first EqualityGoalCount objectives exactly.	fgoalattain	optimoptions only. For optimset, use GoalsExactAchieve
FiniteDifferenceStepSize	Scalar or vector step size factor for finite differences. When you set FiniteDifferenceStepSize to a vector v, forward finite differences steps delta are delta = v.*sign′(x).*max(abs(x),TypicalX); where sign′(x) = sign(x) except sign′(0) = 1. Central finite differences are delta = v.*max(abs(x),TypicalX); Scalar FiniteDifferenceStepSize expands to a vector. The default is sqrt(eps) for forward finite differences, and eps^(1/3) for central finite differences.	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin	optimoptions only. For optimset, use FinDiffRelStep
FiniteDifferenceType	Finite differences, used to estimate gradients, are either 'forward' (the default), or 'central' (centered), which takes twice as many function evaluations but should be more accurate. 'central' differences might violate bounds during their evaluation in fmincon interior-point evaluations if the HonorBounds option is set to false.	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin	optimoptions only. For optimset, use FinDiffType
FunctionTolerance	Termination tolerance on the function value.	fgoalattain, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog	optimoptions only. For optimset, use TolFun

Продовження табл. А.1

1	2	3	4
HessianApproximation	Method of Hessian approximation: 'bfgs', 'lbfgs', {'lbfgs',Positive Integer}, or 'finite-difference'. Ignored when HessianFcn or HessianMultiplyFcn is nonempty.	fmincon	optimoptions only. For optimset, use Hessian
HessianFcn	Function handle to a user-supplied Hessian (see Including Hessians).	fminconfminunc	optimoptions only. For optimset, use HessFcn
HessianMultiplyFcn	Handle to a user-supplied Hessian multiply function. Ignored when HessianFcn is nonempty.	fmincon, fminunc, quadprog	optimoptions only. For optimset, use HessMult
Heuristics	Algorithm for searching for feasible points (see Heuristics for Finding Feasible Solutions): 'none' 'rss' 'round' 'rins'	intlinprog	optimoptions only
HeuristicsMaxNodes	Strictly positive integer that bounds the number of nodes intlinprog can explore in its branch-and-bound search for feasible points. See Heuristics for Finding Feasible Solutions.	intlinprog	optimoptions only
HonorBounds	The default true ensures that bound constraints are satisfied at every iteration. Turn off by setting to false.	fmincon	optimoptions only. For optimset, use AlwaysHonorConstraints
IntegerPreprocess	Types of integer preprocessing (see Mixed-Integer Program Preprocessing): 'none' — Use very few integer preprocessing steps. 'basic' — Use a moderate number of integer preprocessing steps. 'advanced' — Use all available integer preprocessing steps.	intlinprog	optimoptions only

1	2	3	4
IntegerTolerance	Real from 1e-6 through 1e-3, where the maximum deviation from integer that a component of the solution x can have and still be considered an integer. IntegerTolerance is not a stopping criterion.	intlinprog	optimoptions only
JacobianMultiplyFcn	User-defined Jacobian multiply function. Ignored unless SpecifyObjectiveGradientis true for fsolve, lsqcurvefit, and lsqnonlin.	fsolve, lsqcurvefit, lsqlin, lsqnonlin	
LPMaxIterations	Strictly positive integer, the maximum number of simplex algorithm iterations per node during the branch-and-bound process.	intlinprog	optimoptions only
LPOptimalityTolerance	Nonnegative real where reduced costs must exceed LPOptimalityTolerancefor a variable to be taken into the basis.	intlinprog	optimoptions only
MaxFunctionEvaluations	Maximum number of function evaluations allowed.	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin	optimoptions only. For optimset, use MaxFunEvals
MaxIterations	Maximum number of iterations allowed.	All but fzero and lsqnonneg	optimoptions only. For optimset, use MaxIter
MaxFeasiblePoints	Strictly positive integer. intlinprog stops if it finds MaxFeasiblePointsinteger feasible points.	intlinprog	optimoptions only
MaxNodes	Strictly positive integer that is the maximum number of nodes the solver explores in its branch-and-bound process.	intlinprog	
MaxTime	Maximum amount of time in seconds allowed for the algorithm.	intlinprog, linprog	
NodeSelection	Choose the node to explore next. 'simplebestproj' — Best projection. See Branch and Bound. 'minobj' — Explore the node with the minimum objective function. 'mininfeas' — Explore the node with the minimal sum of integer infeasibilities. See Branch and Bound.	intlinprog	optimoptions only

1	2	3	4
ObjectiveCutOff	Real greater than -Inf. The default is Inf.	intlinprog	optimoptions only
ObjectiveImprovementThreshold	Nonnegative real. intlinprog changes the current feasible solution only when it locates another with an objective function value that is at least ObjectiveImprovementThreshold lower: (fold – fnew)/(1 + fold) > ObjectiveImprovementThreshold.	intlinprog	optimoptions only
ObjectiveLimit	If the objective function value goes below ObjectiveLimit and the iterate is feasible, then the iterations halt.	fmincon, fminunc, quadprog	
OptimalityTolerance	Termination tolerance on the first-order optimality.	fgoalattain, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, linprog (interior-point only), lsqcurvefit, lsqlin, lsqnonlin, quadprog	optimoptions only. For optimset, use TolFun
OutputFcn	Specify one or more user-defined functions that the optimization function calls at each iteration. See Output Function or intlinprog Output Functions and Plot Functions.	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, intlinprog, lsqcurvefit, lsqnonlin	
PlotFcn	Plots various measures of progress while the algorithm executes. Select from predefined plots or write your own. @optimplotx plots the current point @optimplotfunccount plots the function count @optimplotfval plots the function value @optimplotconstrviolation plots the maximum constraint violation @optimplotresnorm plots the norm of the residuals @optimplotfirstorderopt plots the first-order of optimality @optimplotstepsize plots the step size @optimplotmilp plots the gap for mixed-integer linear programs See Plot Functions or intlinprog Output Functions and Plot Functions.	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, intlinprog, lsqcurvefit, lsqnonlin. See the individual function reference pages for the values that apply.	optimoptions only. For optimset, use PlotFcns

1	2	3	4
RelativeGapTolerance	Real from 0 through 1. intlinprog stops if the relative difference between the internally calculated upper (U) and lower (L) bounds on the objective function is less than or equal to RelativeGapTolerance: (U – L) / (abs(U) + 1) <= RelativeGapTolerance. intlinprog automatically modifies the tolerance for large L magnitudes: tolerance = min(1/(1+|L|), RelativeGapTolerance)	intlinprog	optimoptions only
RootLPAlgorithm	Algorithm for solving linear programs: 'dual-simplex' — Dual simplex algorithm 'primal-simplex' — Primal simplex algorithm	intlinprog	optimoptions only
RootLPMaxIterations	Nonnegative integer that is the maximum number of simplex algorithm iterations to solve the initial linear programming problem.	intlinprog	optimoptions only
ScaleProblem	For fmincon interior-point and sqp algorithms, true causes the algorithm to normalize all constraints and the objective function by their initial values. Disable by setting to the default false.	fmincon	
Simplex Use Algorithm instead	If 'on', function uses the simplex algorithm.	linprog	optimset only
SpecifyConstraintGradient	User-defined gradients for the nonlinear constraints.	fgoalattain, fmincon, fminimax	optimoptions only. For optimset, use GradConstr
SpecifyObjectiveGradient	User-defined gradients or Jacobians for the objective functions.	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin	optimoptions only. For optimset, use GradObj or Jacobian
StepTolerance	Termination tolerance on x.	All functions except linprog and lsqlin, and the quadprog 'active-set' algorithm	optimoptions only. For optimset, use TolX

Закінчення табл. А.1

1	2	3	4
SubproblemAlgorithm	Determines how the iteration step is calculated.	fmincon	
TypicalX	Array that specifies typical magnitude of array of parameters x. The size of the array is equal to the size of x0, the starting point. Primarily for scaling finite differences for gradient estimation.	fgoalattain, fmincon, fminimax, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonli, quadprog	
UseParallel	When true, applicable solvers estimate gradients in parallel. Disable by setting to false.	fgoalattain, fmincon, fminimax, fminunc, fsolve, lsqcurvefit, lsqnonlin.	

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Малярець Людмила Михайлівна
Ковальова Катерина Олександрівна

ДОСЛІДЖЕННЯ ОПЕРАЦІЙ ТА МЕТОДИ ОПТИМІЗАЦІЇ

Лабораторний практикум в середовищі MATLAB

Самостійне електронне текстове мережеве видання

Відповідальний за видання *Л. М. Малярець*

Відповідальний редактор *М. М. Оленич*

Редактор *Н. І. Ганцевич*

Коректор *Т. А. Маркова*

План 2018 р. Поз. № 13-ЕНП. Обсяг 123 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.