МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Одеська національна академія зв'язку ім. О.С.Попова

Кафедра інформаційних технологій

Л.М. Буката, Л.В. Глазунова

ЧИСЕЛЬНІ МЕТОДИ ТА МОДЕЛЮВАННЯ НА ЕОМ

ЧАСТИНА 1

Навчальний посібник

МОДУЛЬ 2

Наближення функції. Чисельні методи розв'язання диференціальних рівнянь. Методи одновимірної та багатовимірної оптимізації. Математичний пакет MatLab

Одеса 2013

УДК 004.43 План НМВ 2013 р.

Буката Л.Н. Чисельні методи та моделювання на ЕОМ: навчальний посібник. – Ч. 1. – Модуль 2 / Буката Л.Н., Глазунова Л.В. – Одеса: ОНАЗ ім. О.С. Попова, 2013. – 84 с.

Навчальний посібник містить короткі теоретичні відомості щодо методів розв’язання інженерних і наукових задач з використанням чисельних методів. У першій частині посібника наведено методи, алгоритми та приклади розв’язання таких задач, як диференціальні рівняння, наближення функції, оптимізація функції однієї та багатьох змінних.

Навчальний посібник призначено для здобуття теоретичних і практичних знань студентами, які вивчають дисципліни “Програмування інженерних задач”, “Чисельні методи та моделювання на ЕОМ”, “Чисельні методи”, а також буде корисним аспірантам і науковим співробітникам, яким необхідно розв’язувати наукові задачі з використанням чисельних методів і математичних пакетів.

СХВАЛЕНО
на засіданні кафедри
інформаційних технологій
та рекомендовано до друку.
Протокол № 7
від 08 лютого 2013 р.

ЗАТВЕРДЖЕНО
Методичною радою
академії зв'язку.
Протокол № 3/14
від 09.04.2013 р.

ПЕРЕДМОВА

Дисципліна **“**Чисельні методи та моделювання на ЕОМ” викладається в семестрах 2.3 і 2.4 студентам 2-го курсу, які навчаються за напрямом бакалаврської підготовки “Автоматизація та комп’ютерно-інтегровані технології”, призначена для формування знань і навичків з використання комп’ютерів у професійній та повсякденній діяльності, а саме: програмування та розв’язання інженерних задач автоматизації технологічних процесів з використанням чисельних методів і математичних пакетів. Методи та алгоритми розв’язання інженерних задач, наведені у посібнику, також можуть використовувати студенти, які вивчають дисципліну “Програмування інженерних задач” (напрям “Мережі та системи поштового зв’язку” та “Телекомунікації”)), “Чисельні методи” (напрям “Системи технічного захисту інформації”) та “Основи математичного моделювання” (напрям “Телекомунікації”).

Мета дисципліни – формування у студентів знань і навичків з таких областей:

- моделювання технологічних об’єктів;
- математичні пакети;
- чисельне диференціювання та інтегрування;
- розв’язання нелінійних рівнянь;
- розв’язання систем алгебраїчних рівнянь;
- чисельні методи розв’язання диференціальних рівнянь і систем;
- інтерполяція та апроксимація функцій;
- чисельні методи розв’язання задач одновимірної оптимізації функцій;
- розв’язання задач багатовимірної оптимізації.

Програма курсу складається з двох модулів:

Модуль 1 Чисельне обчислення функцій, характеристик матриць і розв’язування нелінійних рівнянь і систем рівнянь.

Модуль 2 Наближення функції. Чисельні методи розв’язання диференціальних рівнянь. Методи одновимірної та багатовимірної оптимізації. Математичний пакет MatLab.

ТЕМА 1. НАБЛИЖЕННЯ ФУНКЦІЙ ДЛЯ МОДЕЛЮВАННЯ: ВИДИ АПРОКСИМУЮЧИХ ФУНКЦІЙ, ЗАСОБИ НАБЛИЖЕННЯ ФУНКЦІЙ. МЕТОДИ ІНТЕРПОЛЯЦІЇ ФУНКЦІЇ

1.1 Поняття апроксимації та інтерполяції

Апроксимація (лат. approximation – наближення) – наближене вираження одних математичних об'єктів іншими, простішими, наприклад, кривих ліній – ламаними, ірраціональних чисел – раціональними, неперервних функцій – многочленами і т. д.

Апроксимація функції $f(x)$ – це побудова такої функції $g(x)$, яка приблизно (у певному сенсі) дорівнює вихідній функції на розглянутому відрізку, тобто $g(x) \cong f(x)$ для $x \in [x_0, x_n]$. Тобто, апроксимацією називають наближення функції $f(x)$ більш простою функцією $g(x)$.

x	$f(x)$
x_0	f_0
x_1	f_1
…	…
x_n	f_n

Близькості цих функцій домагаються шляхом уведення до апроксимуючої функції $g(x)$ вільних параметрів a_0, a_1, …, a_n та відповідних їхньому вибору. Наприклад, за таблицею значень вимірів (x_i, f_i) або обчислень $f(x)$ підібрати (побудувати) функціональну залежність – апроксимуючу функцію $g(x, a_0, a_1, \ldots, a_n)$, графік якої проходить порівняно близько щодо вузлів базової таблиці. Вибрані значення x_i $(i = 0, 1, \ldots, n)$ називаються *вузлами таблиці*, частіше не рівновіддаленими.

Порядок наближення функції:

1) визначити клас шуканої наближеної функції;
2) визначити ступінь близькості вихідної функції та наближеної.

Отже, треба говорити про наближення у певному класі функцій. Наприклад, про наближення функції у класі алгебраїчних багаточленів

$$P_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n,$$

або у класі тригонометричних поліномів

$$Q_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx),$$

що доцільно, якщо $f(x)$ – періодична функція, або у класі узагальнених поліномів

$$G_n(x) = a_0 f_0(x) + a_1 f_1(x) + \ldots + a_n f_n(x),$$

де $f_k(x)$ $(k=0,1,2,\ldots,n)$ – дана система функцій.

Після вибору класу функцій, в якому шукають наближену функцію $g(x)$, потрібно обговорити в якому сенсі вона повинна найліпшим чином наближати функцію $f(x)$ на відрізку $x \in [x_0, x_n]$, тому що слова «найліпшим чином» можна розуміти по-різному, якщо по-різному вибирати ступінь наближення $g(x)$ до $f(x)$. Зокрема, в залежності від мети наближення можна говорити, що функція $g(x)$ найліпшим чином наближує функцію $f(x)$ на відрізку $[x_0, x_n]$ у кожному з цих випадків:

1. Якщо $g(x)$ збігається зі значеннями $f(x)$ у вузлових точках $x_0, x_1, \ldots, x_n$ на проміжку $[x_0, x_n]$, тобто

$f(x_i) = g(x_i)$, $x_i \in [x_0, x_n]$ $(i = 0,1,2,\ldots, n)$ – задача *інтерполяції*.

Інтерполяція – (від лат. interpolatio – перетворення) в обчислювальній математиці спосіб, за допомогою якого за таблицею, що містить певні числові дані, можна знайти проміжні результати, яких нема безпосередньо у таблиці. Наприклад, визначення функції $f(x)$ для аргументів $x \in [x_0, x_n]$ за відомими значеннями $f(x_i)$, де $i = 0, 1, \ldots, n$. Якщо x лежить зовні інтервалу $[x_0, x_n]$, аналогічна процедура називається *екстраполяцією*. Найпростішою є лінійна інтерполяція, за якою приріст функції вважають пропорційним приросту аргументу.

2. Якщо середнє квадратичне відхилення значень $g(x)$ від значень $f(x)$ у вузлових точках $x_0, x_1, \ldots, x_n$ на проміжку $[x_0, x_n]$, тобто величина

$$\Delta = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n}\left(f(x_i) - g(x_i)\right)^2} = \min$$

повинна бути мінімальною у даному класі функцій $g(x)$ – задача *квадратичного наближення* (квадратична апроксимація функції).

3. Якщо у будь-якій точці відрізка $[x_0, x_n]$ функція $g(x)$ відрізняється від $f(x)$ на величину, абсолютне значення якої не більш заданого числа ε, тобто

$\Delta = \max\left|f(x_i) - g(x_i)\right| < \varepsilon$ – задача *рівномірного наближення*.

У задачах теорії коливань, електродинаміки, твердотілої електроніки широко використовуються апроксимації функцій для описання фізичних параметрів середовищ, для задавання характеристик активних і пасивних елементів шляхом радіотехнічних ланцюгів і т.д. В обчислювальній математиці апроксимація функцій є основою для розроблення багатьох методів і алгоритмів.

1.2 Алгебраїчна інтерполяція

Нехай задано функцію $f(x)$: $f(x_0) = y_0$; $f(x_1) = y_1$; …; $f(x_n) = y_n$. Треба побудувати поліном

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

такий, щоб його значення $P_n(x_i) = y_i$, для $i = 1,\ldots, n$ збігалося зі значеннями y_i у вузлах інтерполяції $x_0, x_1, \ldots, x$:

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1 \\ \ldots\ldots\ldots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n \end{cases}$$

Система лінійних алгебраїчних рівнянь (СЛАР) щодо вільних параметрів a_i має розв'язання, оскільки визначник системи є відмінним від нуля, якщо серед вузлів x_i немає однакових. Визначник системи називають визначником Вандермонда.

Визначник Вандермонда: $W = \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ . & . & . & & . \\ 1 & x_n & x_n^2 & & x_n^n \end{vmatrix}$

Систему можна записати у матричному вигляді:

$$\mathbf{W * A = Y}$$

Інтерполяція поліномами має такі переваги, як простота обчислень їхніх значень, диференціювання та інтегрування.

1.3 Лінійна інтерполяція

При лінійній інтерполяції функція замінюється ламаною лінією (рис. 1.1).

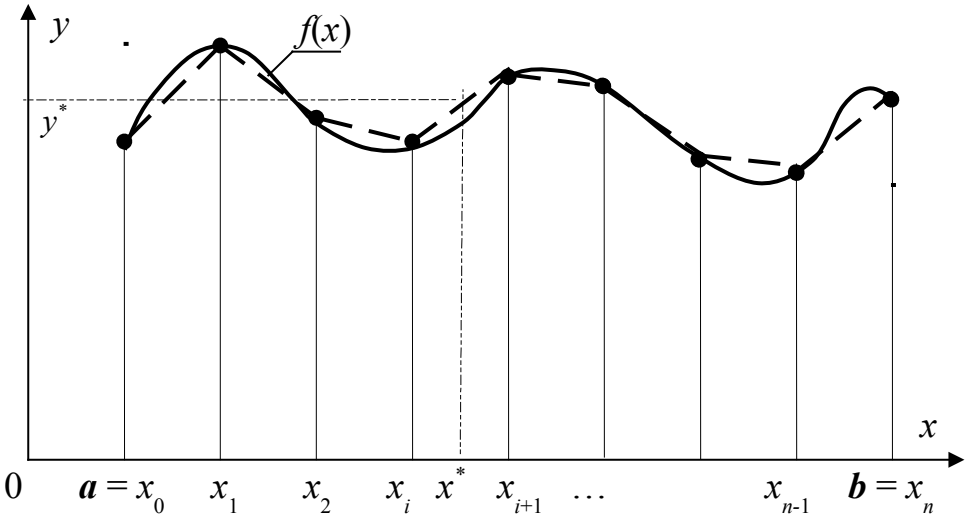


Рисунок 1.1 −Лінійна інтерполяція функції

Формула лінійної інтерполяції:

$$y^* = y_i + (y_{i+1} - y_i)\frac{x^* - x_j}{x_i - x_j},$$

де x^* належить проміжку $[x_i, x_{i+1}]$, $i = 0, 1, \dots, n$.

1.4 Інтерполяційний поліном Лагранжа

Задано функцію $f(x)$: $f(x_0) = y_0$, $f(x_1) = y_1$, …, $f(x_n) = y_n$. Треба побудувати многочлен $g(x)$, де x – довільне значення інтервалу $[x_0, x_n]$.

Формула полінома Лагранжа має вигляд

$$g(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)},$$

або у скороченому вигляді

$$g(x) = \sum_{i=0}^{n} y_i \frac{\prod_{j=0, j\neq i}^{n} (x - x_j)}{\prod_{j=0, j\neq i}^{n} (x_i - x_j)} = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

Для обчислення полінома (многочлена) Лагранжа не треба попереднього визначення коефіцієнтів полінома шляхом розв'язання системи рівнянь, проте для кожного значення x^* поліном доводиться перераховувати знову. Тобто практичне застосування полінома Лагранжа виправдане лише у випадку, якщо інтерполяційна функція обчислюється у порівняно невеликій кількості значень x. Алгоритм методу Лагранжа показано на рис. 1.2.

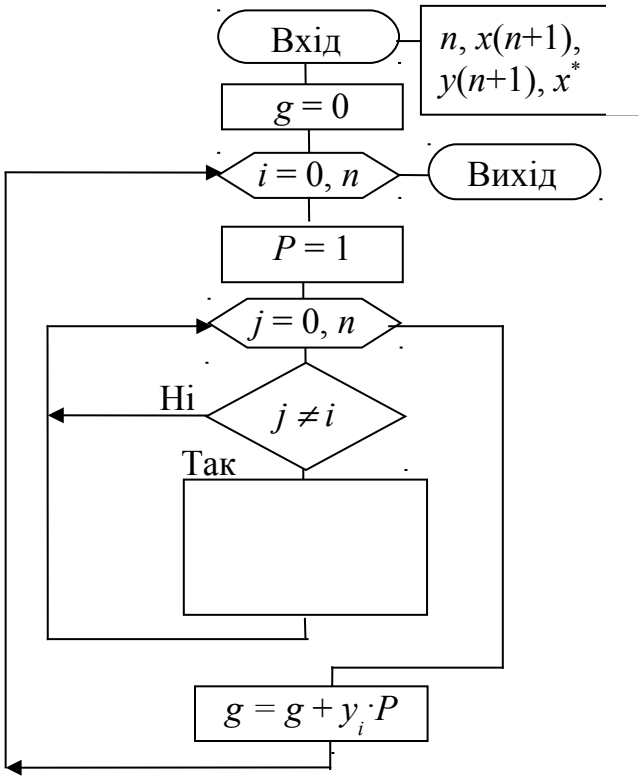


Рисунок 1.2 – Блок-схема алгоритму методу Лагранжа

ТЕМА 2. АПРОКСИМАЦІЯ ТАБЛИЧНИХ ДАНИХ. МЕТОД НАЙМЕНШИХ КВАДРАТІВ. АПРОКСИМАЦІЯ ПОЛІНОМАМИ.

2.1 Апроксимація табличних даних

У процесі вивчення різних питань природознавства, економіки і техніки, соціології, педагогіки доводиться на основі великої кількості дослідних даних виявляти суттєві фактори, які впливають на досліджуваний об'єкт, а також встановлювати форму зв'язку між різними зв'язаними одна з одною величинами (ознаками).

Нехай у результаті досліджень дістали таку таблицю деякої функціональної залежності:

x_i	x_1	x_2	…	x_n
y_i	y_1	y_2	…	y_n

Треба знайти аналітичний вигляд функції y_i, яка добре відображала б цю таблицю дослідних даних. Функцію y_i можна шукати у вигляді інтерполяційного полінома. Але інтерполяційні поліноми не завжди добре відображають характер поведінки таблично заданої функції. До того ж значення y_i дістають у результаті експерименту, а вони, як правило, сумнівні. В такому разі задача інтерполювання табличної функції втрачає сенс. Тому шукають таку функцію $g(x)$, значення якої при $x = x_i$ досить близькі до табличних значень y_i,.., y_n. Формулу $y_i = g(x_i)$ називають *емпіричною* або *рівнянням регресії* y_i на x_i. Емпіричні формули мають велике практичне значення, вдало підібрана емпірична формула дає змогу не тільки апроксимувати сукупність експериментальних даних, «згладжуючи» значення величин y_i, а й екстраполювати знайдену залежність на інші проміжки значень x.

Процес побудови емпіричних формул складається з двох етапів: встановлення загального виду цієї формули і визначення найкращих її параметрів.

Щоб встановити вигляд емпіричної формули, на площині будують точки з координатами (x_i, y_i), де $i = 1,2…,n$. Деякі з цих точок сполучають плавною кривою, яку проводять так, щоб вона проходила якомога ближче до всіх даних точок. Після цього візуально визначають, графік якої з відомих нам функцій найкраще підходить до побудованої кривої. Звичайно, намагаються підібрати найпростіші функції: лінійну, квадратичну, дробово-раціональну, степеневу, показникову, логарифмічну.

Встановивши вигляд емпіричної формули, треба знайти її параметри (коефіцієнти). Найточніші значення коефіцієнтів емпіричної формули визначають *методом найменших квадратів*. Цей метод запропонували відомі математики К. Гаусс і А. Лежандр.

2.2 Метод найменших квадратів. Апроксимація поліномами

Нехай задана функція $f(x)$: $f(x_0) = y_0$; $f(x_1) = y_1$; …; $f(x_n) = y_n$. Треба побудувати багаточлен $g_m(x)$ (обов'язково $m<n$) такий, щоб сума квадратів (тоді не треба враховувати знак) відхилень була мінімальна

$$S = \sum_{i=0}^{n} \left(g_m(x_i) - f(x_i)\right)^2 \to \min, \qquad (2.1)$$

тоб то необхідно знайти екстремум (мінімум) суми крадратів (максимум нас не цікавить). Якщо функція $g_m(x)$ є алгебраїчний багаточлен

$$g_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^n,$$

то умова наближення (2.1) має вигляд:

$$S = \sum_{i=0}^{n} \left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^n - y_i\right)^2 \to \min \qquad (2.2)$$

Необхідна умова мінімуму функції багатьох змінних (2.2)─ її частинні похідні мають дорівнювати нулю, тобто

$$\frac{\partial S}{\partial a_1} = 0,\ \frac{\partial S}{\partial a_2} = 0,\ldots,\ \frac{\partial S}{\partial a_m} = 0;$$

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=0}^{n} \left(a_0 + a_1 x_i + \ldots + a_m x_i^n - y_i\right)\cdot 1 = 0 \\ \dfrac{\partial S}{\partial a_1} = 2\sum\limits_{i=0}^{n} \left(a_0 + a_1 x_i + \ldots + a_m x_i^n - y_i\right)\cdot x_i = 0 \\ \ldots\ldots\ldots\ldots\ldots \\ \dfrac{\partial S}{\partial a_m} = 2\sum\limits_{i=0}^{n} \left(a_0 + a_1 x_i + \ldots + a_m x_i^n - y_i\right)\cdot x_i^m = 0 \end{cases} \qquad (2.3)$$

Ця система лінійних алгебраїчних рівнянь визначає усі коефіцієнти a_i. Система (2.3) називається системою *нормальних рівнянь*. Після рівносильних перетворень системи (2.3) маємо систему:

$$\begin{cases} a_0\sum\limits_{i=0}^{n} x_i^0 + a_1\sum\limits_{i=0}^{n} x_i^1 + \ldots + a_m\sum\limits_{i=0}^{n} x_i^m = \sum\limits_{i=0}^{n} y_i \\ a_0\sum\limits_{i=0}^{n} x_i^1 + a_1\sum\limits_{i=0}^{n} x_i^2 + \ldots + a_m\sum\limits_{i=0}^{n} x_i^{m+1} = \sum\limits_{i=0}^{n} y_i x_i \\ \ldots\ldots\ldots\ldots \\ a_0\sum\limits_{i=0}^{n} x_i^m + a_1\sum\limits_{i=0}^{n} x_i^{m+1} + \ldots + a_m\sum\limits_{i=0}^{n} x_i^{m+m} = \sum\limits_{i=0}^{n} y_i x_i^m \end{cases} \qquad (2.4)$$

Систему нормальних рівнянь (2.4) можно записати за допомогою матриць:

S*A=B; де **A**=$(a_0, a_1, \ldots, a_m)$; $S = \left\{S_{kj}\right\}_{kj=0}^{m} = \sum\limits_{i=0}^{n} x_i^{k+j}$; $B = \left\{b_k\right\}_{k=0}^{m} = \sum\limits_{i=0}^{n} y_i x_i^k$. (2.5)

Апроксимація лінійним поліномом

Нехай між табличними даними (x_i,y_i $i = 1,2\ldots,n$) існує лінійна залежність. Шукатимемо емпіричну формулу у вигляді $y = ax + b$, де коефіцієнти a і b невідомі.

Знайдемо значення a і b, за яких функція $S(a,b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$ матиме мінімальне значення. Щоб знайти ці значення, прирівняємо до нуля часткові похідні функції $S(a,b)$

$$\begin{cases} \frac{\partial S}{\partial a} = 2\sum_{i=1}^{n} (y_i - ax_i - b)(-x_i) = 0, \\ \frac{\partial S}{\partial b} = 2\sum_{i=1}^{n} (y_i - ax_i - b)(-1) = 0. \end{cases}$$

Звідси, врахувавши, що $\sum_{i=1}^{n} b = nb$, маємо

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i, \\ a\sum_{i=1}^{n} x_i + nb = \sum_{i=1}^{n} y_i. \end{cases}$$

Розв'язавши відносно a і b останню систему, знайдемо

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}, \qquad b = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}.$$

Якщо ми маємо дві точки (x_0, y_0) та (x_1, y_1), то рівняння прямої має вигляд (рис. 2.2):

$$y = y_0; \ \frac{x - x_0}{x_0 - x_1} + y\frac{x - x_0}{x_1 - x_0}.$$

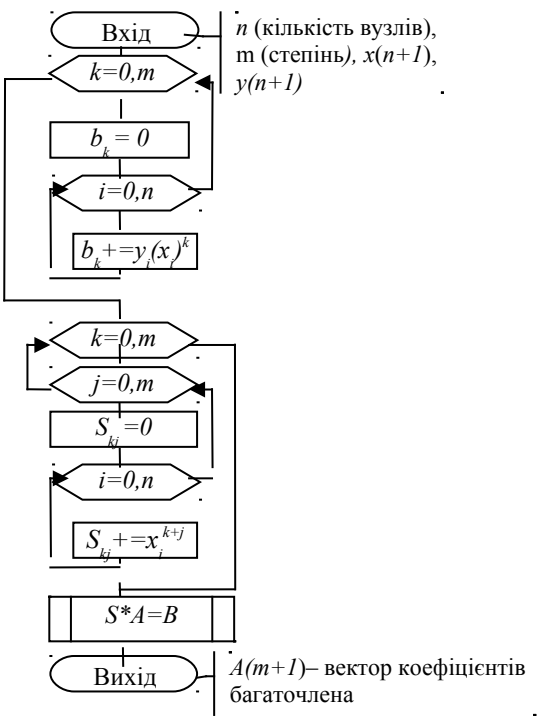


Рисунок 2.1– Блок-схема методу НК

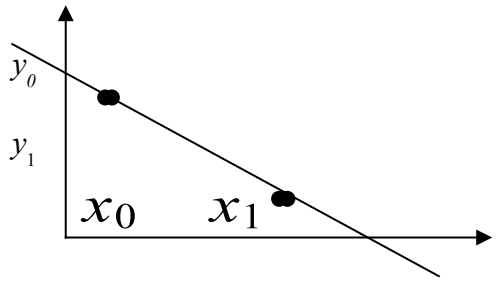


Рисунок 2.2 – Лінійна апроксимація

ТЕМА 3. ОПИС ПРОЦЕСІВ ДИФЕРЕНЦІАЛЬНИМИ РІВНЯННЯМИ. ОБЧИСЛЮВАЛЬНІ МЕТОДИ РОЗВ'ЯЗАННЯ ЗВИЧАЙНИХ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ ТА ЇХНІХ СИСТЕМ: МЕТОДИ ЕЙЛЕРА ТА РУНГЕ-КУТИ

3.1 Постановка задачі

Звичайні диференціальні рівняння (ЗДР) широко використовуються для математичного моделювання процесів і явищ у різних галузях науки і техніки. Перехідні процеси у радіотехніці, кінетика хімічних реакцій, динаміка біологічних популяцій, рух космічних об'єктів, моделі економічного розвитку досліджуються за допомогою ЗДР.

Диференціальне рівняння n-го порядку має вигляд:

$$y^{(n)} = f(x, y, y', \ldots\ y^{(n-1)})$$

або $$\varphi(x, y, y', \ldots\ y^{(n)}) = 0. \qquad (3.1)$$

де невідомими є функція $y(x)$ та її n похідних від x (y', ... $y^{(n)}$).

З теорії ЗДР відомо, що рівняння (3.1) є еквівалентне до *системи n рівнянь першого порядку*

$$\varphi_k(x, y_1, y_1', y_2, y_2', \ldots, y_n, y_n') = 0\ , \quad \text{де}\ k = 1, \ldots, n. \qquad (3.2)$$

Рівняння (3.1) і відповідна йому система (3.2) мають нескінченну множину розв'язань. Єдинє розв'язання виділяють за допомогою додаткових умов, яким мають задовольняти рівняння. Залежно від різновиду таких умов розглядають три типи задач, для яких доведено існування й одиничність розв'язання.

Перший тип – це задачі Коші або задачі з початковими умовами. Для таких задач крім рівняння (3.1) у деякій точці x_0 мають бути задані початкові умови, тобто значення функції $y(x)$ та її похідних

$$y(x_0) = y_0,\ \ y'(x_0) = y_{10},\ \ldots,\ y^{(n-1)}(x_0) = y_{n-1,0}.$$

Для системи ЗДР вигляду (3.2) початкові умови задаються у вигляді

$$y_1(x_0) = y_{10}, y_2(x_0) = y_{20}, \ldots, y_n(x_0) = y_{n0}. \qquad (3.3)$$

Для ЗДР першого порядку задача Коші має вигляд:

$$y'(x) = f(x, y),\ x_0 \le x \le x_k,\ \ y(x_0) = y_0.$$

До чисельних методів розв'язання задачі Коші відносяться методи Рунге-Кути, метод малого параметра та інші.

До *другого типу задач* відносяться так звані граничні або крайові задачі, для яких додаткові умови задаються у вигляді функціональних співвідношень між шуканими розв'язками. Кількість умов має збігатися з порядком n рівняння або системи. Якщо розв'язання задачі визначено на інтервалі $x \in [x_0, x_k]$, то такі умови може бути задано як на межах, так і всередині інтервалу. Мінімальний порядок ЗДР, для яких може бути сформульовано граничну задачу, дорівнює двом. Прикладом такої задачі є задача знаходження статичного прогину

навантаженої струни з закріпленими кінцями: $y''(x) = -f(x)$, $a \le x \le b$, $y(a)=y(b)=0$, де $f(x)$ – зовнішнє згинальне навантаження на одиницю довжини струни, яке поділене на пружність струни. До наближенних методів розв'язання крайових задач відносяться розкладання у ряд Фур'є, методи Рітца та Галеркіна.

Третій тип задач для ЗДР – це задачі на власні значення. Такі задачі відрізняються тим, що крім шуканих функцій $y(x)$ та їхніх похідних до рівняння входять додатково q невідомих параметрів λ_1, λ_2, …, λ_q, які називаються власними значеннями. Задача на власні значення має вигляд

$y'(x) = f(x,y;\lambda_1,\lambda_2,\ldots,\lambda_q)$, $y = \{y_1,y_2,\ldots,y_n\}$, $f = \{f_1, f_2,\ldots, f_n\}$, число крайових умов дорівнює $n+q$.

Для єдиного розв'язку на інтервалі $[x_0, x_k]$ треба задати $q + n$ крайових умов. Як приклад, можна назвати задачі визначення власних частот, структури електромагнітних полів і механічних напруг у коливальних системах, задачі обчислення фазових коефіцієнтів (коефіцієнтів загасання) розподілу напруги полів хвильових процесів тощо. Приклад задачі на власні значення – визначення власних коливань струни:

$\frac{d}{dx}[k(x)\frac{du}{dx}] + \lambda \rho(x) \cdot u(x) = 0$, де $k(x)$ та $\rho(x)$ – жорсткість і щільність струни, $u(x)$ – профіль коливань. Початкові умови задачі: $u(0) = u(1) = 0$, $u'(0) = 1$, тобто дві крайові та додаткова для визначення λ.

До *чисельного розв'язання ЗДР* доводиться звертатися, коли не вдається здобути аналітичний розв'язок задачі через відомі функції. Хоча для деяких задач чисельні методи виявляються більш ефективними навіть за наявності аналітичних розв'язків.

Більшість методів *розв'язання* ЗДУ засновано на задачі Коші, алгоритми й програми для якої буде далі розглянуто.

3.2 Методи Рунге-Кути

Розгляд чисельних методів *розв'язання* диференціальних рівнянь, розпочнемо з їхньої важливої категорії, відомої під загальною назвою методів Рунге-Кути. Названі на честь німецьких математиків Карла Рунге і Мартіна Кути, які відкрили ці методи.

Методи Рунге-Кутти мають такі *властивості*:

✓ Ці методи є одноступінчастими: щоб обчислити $y_{m+1}(x_{m+1})$, потрібна інформація про попередню точку (x_m, y_m), де m − номер ітерації обчислень.

✓ Ці методи майже збігаються з методом Тейлора аж до членів порядку h^p, де степінь p − є порядковий номер або порядок методу (відрізняється для різних методів).

✓ Ці методи не вимагають обчислення похідних від $f(x, y)$, а вимагають обчислення самої функції.

Спочатку розглянемо *геометричну побудову* й виведемо деякі формули на основі геометричних аналогій, після чого підтвердимо здобуті результати аналітично (рис. 3.1).

Припустимо, що відомо точку A с координатами (x_m, y_m) на шуканій кривій. Тоді можна провести пряму лінію з тангенсом кута нахилу $y'_m = f(x_m, y_m)$, яка пройде через точку (x_m, y_m). На рис. 3.1 показана крива $y(x)$, яку треба знайти, пряма лінія L_1 побудована так, як це тільки що описано. Тоді наступною точкою розв'язання можна вважати точку B, де пряма L_1 перетне ординату, проведену через точку $x_{m+1} = x_m + h$.

Рівняння прямої L_1 має вигляд:

$$y = y_m + y'_m(x - x_m), \tag{3.4}$$

оскільки $y' = f(x_m, y_m)$ і, крім того, $x_{m+1} = x_m + h$. Тоді рівняння (3.4) набуде вигляду

$$y_{m+1} = y_m + h*f(x_m, y_m) \tag{3.5}$$

Відхилення розв'язку y_{m+1} в точці $x = x_{m+1}$ від значення функції $y(x_{i+1})$ показано як відрізок ε. Очевидно, знайдене у такий спосіб наближене значення узгоджується з розкладанням у ряд Тейлора аж до членів порядку h, так що похибка розв'язку метода дорівнює $\varepsilon_t = k\,h^2$.

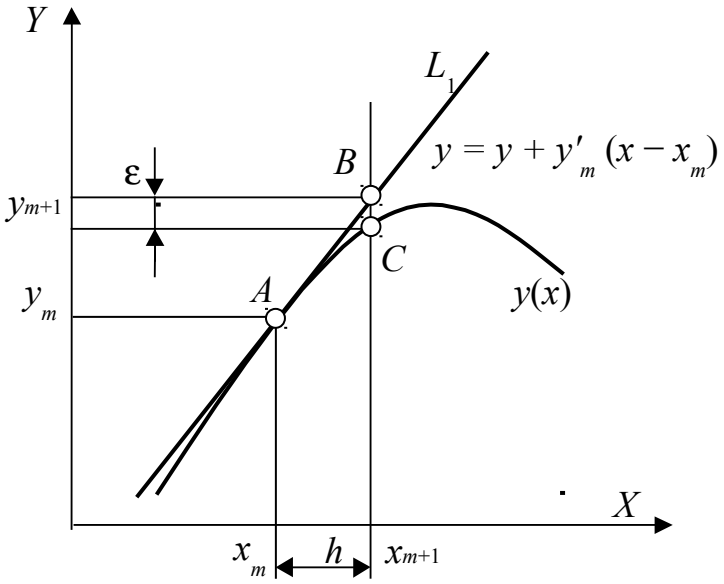


Рисунок 3.1 − Геометрична інтерпретація методу Рунге-Кути першого порядку

Формула (3.5) описує метод Ейлера. Це один з найпоширеніших і відомих методів чисельного інтегрування диференціальних рівнянь. Зазначимо, що цей метод є методом Рунге-Кути першого порядку.

Розглянемо виправлений метод Ейлера. У *виправленому методі Ейлера* знаходимо середній тангенс кута нахилу дотичної для двох точок: (x_m, y_m) та $(x_m + h, y_m + hy'_m)$. Остання точка є та сама, яка у методі Ейлера позначалася $(x_{m+1},$

y_{m+1}). Геометричний процес знаходження точки (x_{m+1}, y_{m+1}) можна простежити за рис. 3.2.

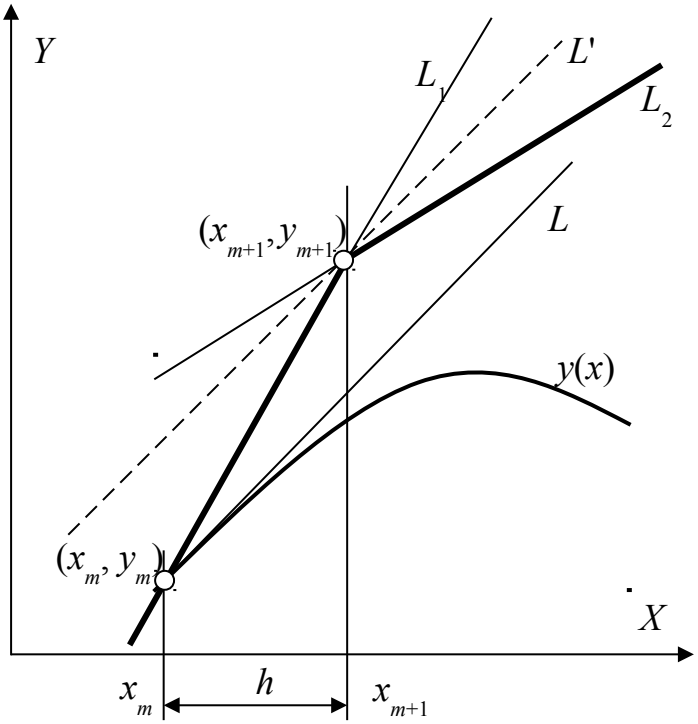


Рисунок 3.2 − Геометрична інтерпретація виправленого методу Ейлера

За допомогою звичайного методу Ейлера будуємо точку ($x_m + h$, $y_m + hy'_m$), яка лежить на прямій L_1. У цій точці знову обчислюється тангенс, який дає пряму L_2. Через середину кута між цих ліній проведемо лінію L'. Нарешті, через точку (x_m, y_m) проводимо пряму L, паралельну до L'. Точка, у якій пряма L перетнеться з ординатою, відновленою з $x_{m+1} = x_m + h$, і буде шуканою точкою (x_{m+1}, y_{m+1}).

Тангенс кута нахилу прямої L' і прямої L дорівнює

$$\Phi\ (x_m, y_m, h) = \frac{1}{2}[f(x_m, y_m) + f(x_m + h, y_m + y'_m h)], \qquad (3.6)$$

де $y'_m = f(x_m, y_m)$. (3.7)

Рівняння лінії L при цьому записується у вигляді

$$y = y_m + (x - x_m)\Phi(x_m, y_m, h),$$

так що $$y_{m+1} = y_m + h\ \Phi\ (x_m, y_m, h). \qquad (3.8)$$

Співвідношення (3.6), (3.7) і (3.8) описують виправлений метод Ейлера. Щоб з'ясувати, наскільки добре цей метод узгоджується з розкладанням у ряд Тейлора, пригадаємо, що розкладання у ряд функції $f(x, y)$ можна записати у такий спосіб:

$$f(x, y) = f(x_m, y_m) + (x - x_m)\partial f/\partial x + (y - y_m)\partial f/\partial y + \ldots, \qquad (3.9)$$

де частинні похідні обчислюються при $x = x_m$ та $y = y_m$.

Підставляючи до формули (3.9) $x = x_m + h$ та $y = y_m + h\,y'_m$ і використовуючи вираз (3.7) для y'_m, отримаємо

$$f(x_m + h, y_m + hy'_m) = f + hf_x + hff_y + O(h^2),$$

де знову функція f та її похідні обчислюються в точці (x_m, y_m). Підставимо результат до (3.6) і зробимо необхідні перетворення, маємо

$$\Phi(x_m, y_m, h) = f + h/2(f_x + ff_y) + O(h^2).$$

Підставимо цей вираз до (3.8) та порівняємо з рядом Тейлора

$$y_{m+1} = y_m + hf + h^2/2(f_x + ff_y) + O(h^3).$$

Очевидно, що виправлений метод Ейлера узгоджується з розкладанням у ряд Тейлора аж до членів степеня h^2, який по суті є *методом Рунге – Кутти другого порядку*.

Методи *Рунге-Кути третього і четвертого* порядків можна вивести аналогічно до того, як це робилося при виведенні методів першого й другого порядків. Не будемо тут відтворювати усі ці викладки, а обмежимося наведенням формул, які описують метод четвертого порядку, один з найуживаніших методів інтегрування диференціальних рівнянь. Цей *класичний метод Рунге – Кутти* описується системою з п’яти співвідношень:

$$y_{m+1} = y_m + h/6(R_1 + 2R_2 + 2R_3 + R_4)\,, \qquad (3.10)$$

де

$$R_1 = f(x_m, y_m);$$
$$R_2 = f(x_m + h/2, y_m + hR_1/2);$$
$$R_3 = f(x_m + h/2, y_m + hR_2/2);$$
$$R_4 = f(x_m + h, y_m + hR_3)\,.$$

Похибка розв’язку для цього методу дорівнює $\varepsilon_t = kh^5$, так що формули (3.10) описують метод четвертого порядку. Зауважимо, що при використанні цього методу функцію треба обчислювати чотири рази.

ТЕМА 4. СИСТЕМА МАТЕМАТИЧНИХ РОЗРАХУНКІВ MatLab

4.1 Елементи математичного пакета MatLab. Особливості програмування у MatLab

Середовище MatLab − це інтерактивна система для виконання інженерних і наукових розрахунків, орієнтована на роботу з масивами даних.

Найбільш відомі напрямки використання системи MatLab:

- математика та обчислення;
- розробка алгоритмів;
- обчислювальний експеримент, імітаційне моделювання;
- аналіз даних, дослідження та візуалізація результатів;
- наукова та інженерна графіка;
- розробка додатків, включаючи графічний інтерфейс користувача.

MatLab містить величезну кількість операторів і функцій для розв'язання різноманітних практичних задач, що знімає потребу написання досить складних програм. Наприклад, це функції обігу та транспонування матриць, обчислення значень похідних та інтегралів тощо. Система MatLab має вхідну мову, що нагадує Бейсик (з домішкою Фортрану та Паскаля). Запис програм у системі є традиційним і звичним для більшості користувачів комп'ютерів. Крім того, система надає можливість редагувати програми за допомогою будь-якого звичного для користувача текстового редактора. Вона має і свій власний редактор з налагоджувачем. Головне вікно MatLab наведено на рис. 4.1. Воно складається (за замовченням − default) з наступних частин: рядок команд меню (File, Edit, View, Window та Help), рядок піктограм, вікна Command Window, Command History, Workspace, Current Directory.

Робочий стіл системи MatLab містить такі інструментальні вікна, частина яких не візуальна при початковому запуску:

- Command Window (Командное Окно) – виконує усі функції та команди системи MatLab;
- Command History (История Команд) – перегляд функцій, введених раніше в Command Window, їх копіювання та виконання;
- Launch Pad (Окно Запуска) – запускає всі інструменти та забезпечує доступ до усіх пакетів системи MatLab;
- Current Directory Browser (Текущий Каталог) – перегляд файлів MatLab, а також виконання таких операцій над файлами, як пошук та відкриття файлів;
- Help Browser (Помощь) – пошук та перегляд документації за усіма функціями та засобами системи MatLab;
- Workspace Browser (Рабочее Пространство) – перегляд та змінення вмісту робочого простору (workspace) системи MatLab;
- Array Editor (Редактор Массивов Данных) – перегляд вмісту масивів даних, записаних у вигляді таблиці та редагування даних;
- Editor/Debugger (Редактор/Отладчик) – для створення, редагування М-файлів, тобто файлів, які містять функції системи MatLab. Кожне

вікно може бути виведене з конфігурації робочого столу натисканням кнопки зі стрілкою у верхньому правому куті вікна.

Вікно Command Window призначене для роботи у командному режимі – виконується тільки одна команда або оператор за форматом:

>> команда (оператор)<Enter>

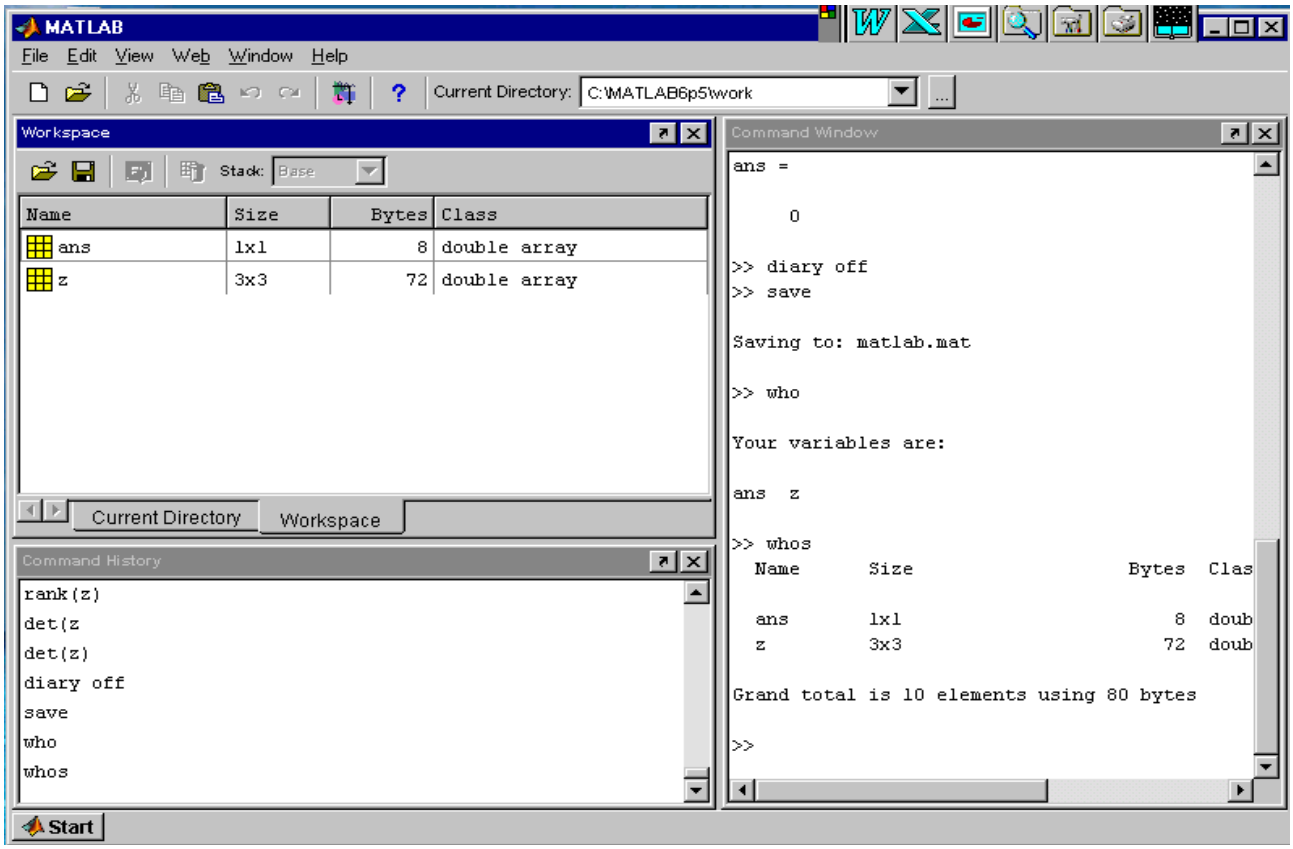


Рисунок 4.1 – Вікно системи MatLab

Символи >> означають запрошення MatLab до введення команди. Після натискання на клавішу <Enter> відразу ж виконується команда або оператор, наприклад:

```
>> who
Your variables are:
ans    z
```

У наведеному прикладі після команди who у наступному рядку виведено імена змінних оперативної пам'яті `ans` та `z` (рис. 4.1). Декілька операторів у командному рядку:

```
>>x = 2; y = x + 7
```

виконують присвоєння значення 2 змінній x та обчислення значення змінної y за формулою y = x + 7; після натискання на клавішу <Enter> отримаємо результат y = 9.

Вікно Command History запам'ятовує та відображає перелік команд, які було введено. Ці команди, після виокремлення, можна копіювати у вікно Command Window та в m-файл. *Вікно Workspace* містить інформацію про змінні

в оперативній пам'яті: Name – ім'я змінної, `Size` – розміри матриці, `Bytes` – кількість байтів, `Class` – тип змінної. *Вікно Current Directory* містить список файлів робочої теки MatLab.

Наведемо деякі корисні команди *керування вікном командного режиму*:

- clc – очищує вікно Command Window і розміщує курсор у його лівому верхньому куті;
- clear – очищує оперативну пам'ять (Workspace) від змінних;
- who – виводить список змінних в оперативній пам'яті;
- save *<file_name>* – зберегти значення змінних з оперативної пам'яті в файл;
- load *<file_name>* – завантажити значення змінних з файла в оперативну пам'ять;
- diary on/off – вмикає/вимикає режим запису протоколу роботи у вікні Command Window до файла;
- type *<file_name>* – виведення на екран файла зі змістом протоколу, який записано командою diary;
- home – повертає курсор до лівого верхнього кута вікна;
- echo *<file_name>* on – вмикає режим виведення на екран тексту Script-файла (файла-сценарія);
- echo *<file_name>* off – вимикає режим виведення на екран тексту Script-файла;
- echo *<file_name>* – змінює режим виведення на протилежний;
- echo on all – вмикає режим виведення на екран тексту усіх m-файлів;
- echo off all – вимикає режим виведення на екран тексту усіх m-файлів;
- more on – вмикає режим посторінкового виведення (корисний при перегляді великих m-файлів)

Починаючи з версії MatLab 6.0 обидві команди clc та home діють аналогічно – очищують екран і розміщують курсор у лівому верхньому куті вікна командного режиму роботи. Команди echo дозволяють вмикати чи вимикати відображення тексту m-файлів за кожного звертання до них. Як правило, відображення тексту файлів сильно захаращує екран і досить часто не є необхідним. За великих розмірів файлів початок їхнього тексту (лістинга) ховається далеко за межами області перегляду (поточного вікна командного режиму). Тому для перегляду довгих лістингів файлів доцільно вмикати посторінковий режим виведення командою more on.

Повідомлення про помилки й виправлення помилок

MatLab діагностує введені команди і вирази та виводить відповідні повідомлення про помилки чи попередження. Розглянемо декілька прикладів помилкових виразів:

```
» 1 / 0
    % ділення на нуль
    Warning: Divide by zero
    ans = Inf
```

```
» sqr(2)
    % корінь квадратний – sqrt, а не sqr
    ??? Undefined function or variable 'sqr'.
```

Для громіздких виразів зручно користуватись редактором. Для цього достатньо після виразу натиснути клавішу *Tab*. Система виведе підказку, проаналізувавши введені символи. Якщо варіантів декілька, клавішу *Tab* доведеться натискати ще раз. Із запропонованих системою операторів вибираємо потрібний і натискаємо клавішу ENTER.

Іноді у ході виведення результатів обчислень виводиться скорочення NaN (від слів **N**ot **a** **N**umber – не число). Воно означає невизначеність, наприклад, виду 0/0 чи Inf/Inf, де Inf – системна змінна зі значенням машинної нескінченості (число 10^{308}).

Взагалі кажучи, у MATLAB слід відрізняти попередження про помилку від повідомлення про неї. Попередження (звичайно після слова Warning) не зупиняють обчислень, а лише попереджують користувача про те, що діагностована помилка здатна вплинути на хід обчислень. Повідомлення про помилку (після знаків ???) зупиняє обчислення.

4.2 Основні об’єкти MatLab

Математичні вирази

Центральним поняттям усіх математичних систем є математичний вираз. Він задає те, що має бути обчислено у чисельному (іноді символьному) вигляді. Приклади простих математичних виразів:

```
2+3
2.301*sin(x)
4+exp(3)/5
sqrt(y)/2
sin(pi/2)
```

Математичні вирази будуються на основі чисел, констант, змінних, операторів, функцій та різних спецзнаків. Коротко пояснимо суть цих понять.

Дійсні та комплексні числа

Числа можуть бути цілими і дійсними з фіксованою або рухомою крапкою. Можливим є подання чисел в експоненціальній формі зі зазначенням мантиси, символу “e” і порядку числа. Приклади подання чисел:

```
0
2
-3
2.301   0.00001   123.456e-24
-234.456e10
```

Комплексні числа містять дійсну та уявну частини. Уявна частина має множник i або j, який означає корінь квадратний з –1 $\left(\sqrt{-1}\right)$:

```
3i
2j
2+3i
-3.141i
-123.456+2.7e-3i
```

Спеціальна функція real(z) повертає дійну частину комплексного числа, а функція imag(z) – уявну. Для отримання модуля комплексного числа використовують функцію abs(z), а для обчислення фази – angle(Z):

```
» i
    ans = 0 +1.0000i
» j
    ans = 0 + 1.0000i
» z=2+3i
    z = 2.0000 + 3.0000i
» abs(z)
    ans = 3.6056
» real(z)
    ans = 2
» imag(z)
    ans = 3
» angle(z)
    ans = 0.9828
```

Взагалі кажучи, у MatLab не прийнято поділяти числа на цілі і дійсні, короткі й довгі тощо, як це прийнято у більшості мов програмування, хоча задавати числа у таких формах можна.

Константи та системні змінні

Константа – це завчасно визначене числове або символьне значення, подане унікальним ім'ям. Числа (наприклад 1, –2 та 1.23) є безіменними *числовими константами.* Інші види констант у MatLab називають *системними змінними,* оскільки, з одного боку, вони задаються системою при її завантаженні, а з іншого – можуть перевизначатися.

Основні *системні змінні*:

pi – число $\pi = 3{,}1416$;

i або j – уявна одиниця $\left(\sqrt{-1}\right)$;

eps – похибка обчислень над числами з рухомою крапкою $eps = 2^{-52}$, що приблизно складає 2,22e–16;

Inf – подання машинної додатної нескінченості, складає 10^{308}. При операціях ділення на нуль, яке призводить до результатів надто великих, їх буде подано у вигляді числа з рухомою крапкою;

ans – результат виконання останньої операції. Змінна ans створюється автоматично, коли не визначено вихідні аргументи якогось оператора;

NaN – зазначає нечисловий характер даних (скорочення від Not a Number – не число);

realmin – найменше нормалізоване додатне число у форматі з рухомою крапкою (2^{-1022});

realmax – найбільше число у форматі з рухомою крапкою (2^{1023}).

Приклади використання системних змінних:

```
» 2*pi
    ans = 6.2832
» cos([0:2*pi])
    ans = 1.0000 0.5403 -0.4161 -0.9900 -0.6536
    0.2837 0.9602
» realmin
    ans = 2.2251e-308
» realmax
    ans = 1.7977e+308
» 1/0
    Warning: Divide by zero
    ans = Inf
» 0/0
    Warning: Divide by zero
    ans = NaN
```

Як зазначалось, системні змінні можуть перевизначатися. Так системній змінній eps можна надати інше значення, наприклад: eps=0.0001. А змінні i та j можна використовувати як індекси у циклах for. Проте це може спричинити плутанину, якщо всередині циклу користувач задасть вирази з комплексними числами. Щоб уникнути цього, можна використовувати як індекси I та J замість i та j.

Текстові коментарі

Оскільки MatLab використовується для достатньо складних обчислень, важливе значення має наочність їхнього описання. Вона досягається, зокрема, за допомогою текстових коментарів. *Текстові коментарі* позначаються символом “%”, наприклад:

```
% Bit is factorial function
```

Зауважимо, що можна, але не бажано вводити неангломовні коментарі, оскільки це інколи може зробити програму недієздатною.

Змінні та надання їм значень

Змінні – це іменовані об’єкти для зберігання числових, символьних, векторних або матричних даних. Для надання змінним значень використовують операцію присвоєння “=”. Типи змінних заздалегідь не декларуються. Вони

визначаються виразом, значення якого присвоюється змінній. Так, якщо цей вираз – вектор або матриця, то змінна буде векторною або матричною.

Ім'я змінної (її ідентифікатор) має бути унікальним, тобто не збігатися з іменами інших змінних, функцій та процедур системи, та може містити до 31 символів: літер, цифр і символу підкреслення "_", але починатися має лише з літери. Для символьних змінних їхні значення записуються в апострофах, наприклад: s = 'Demo'.

Оператори та функції

Оператор – це спеціальне позначення певної операції над даними – операндами. Найпростішими арифметичними операторами є знаки суми "**+**", віднімання "**–**", множення "*****" і ділення "***/***". Оператори використовуються разом з операндами. Наприклад, у виразі 2 + 3 знак "+" є оператором додавання, а числа 2 і 3 – операндами.

Зауважимо, що більшість операторів у MatLab відносяться до матричних операцій, що може спричинити чималі непорозуміння. Наприклад, оператори множення "*****" та ділення "***/***" обчислюють добуток і частку від ділення двох багатовимірних масивів, векторів або матриць. Існує низка спеціальних операторів, наприклад, оператор "****" означає ділення справа наліво, а оператори "**.***" та "***./***" означають відповідно поелементне множення та поелементне ділення масивів.

Пояснимо сказане на прикладах операцій з векторами:

```
» V1 = [2   4   6   8]
    V1 = 2  4  6  8
» V2 = [1   2   3   4]
    V2 = 1  2  3  4
 » V1/ V2
    ans = 2
» V1.*V2
    ans = 2   8   18   32
» V1./ V2
    ans = 2   2   2   2
```

Повний список операторів можна побачити, виконавши команду
» help ops.

Функції можуть бути *вбудованими* (внутрішніми) і *зовнішніми*, які містять свої визначення в m-файлах. Вбудовані функції, наприклад, sin(x), exp(y), зберігаються у відкомпільованому ядрі системи MATLAB, а тому вони виконуються максимально швидко.

Зі списком елементарних функцій можна ознайомитись, виконавши команду help elfun, а зі списком спеціальних функцій – за допомогою команди help specfun.

*Оператор послідовності "***:***"* (двокрапка) використовують для формування упорядкованих числових послідовностей, наприклад, при створенні векторів або для значень абсциси при побудові графіків.

Формат оператора:

початкове_значення **:** *крок* **:** *кінцеве_значення*

Ця конструкція утворює зростаючу послідовність чисел, яка розпочинається від початкового значення, збільшується із заданим кроком і завершується кінцевим значенням. Якщо крок не задано, то він набуває значення 1. Якщо кінцеве значення задано меншим за початкове, – виведеться повідомлення про помилку. Приклади використання:

```
» 1 : 5
ans =
    1   2   3   4   5
» i = 0 : 2 : 10
    i = 0   2   4   6   8   10
» j = 10 : –2 : 2
    j = 10   8   6   4   2
» V = 0 : pi/2 : 2*pi
    V = 0   1.5708   3.1416   4.7124   6.2832
» X = 1 : –.2 : 0
    X = 1.0000   0.8000   0.6000   0.4000   0.2000   0
» 5 : 2
    ans = Empty matrix: 1 – by – 0
```

Як зазначалось, належність MatLab до матричних систем вносить корективи у призначення операторів і за невмілого їхнього використання спричиняє казуси. Розглянемо характерний приклад:

```
» x = 0 : 5
    x = 0 1 2 3 4 5
» cos(x)
    ans = 1.0000 0.5403 –0.4161 –0.9900 –0.6536 0.2837
» sin(x)/x
    ans = –0.0862
```

Обчислення масиву косинусів тут відбулося коректно. А ось обчислення масиву значень функції sin(x)/x дає несподіваний, на перший погляд, ефект – замість масиву з шести елементів обчислено лише одне значення! Причина "парадокса" полягає в тому, що оператор "/" обчислює відношення двох матриць, векторів або багатовимірних масивів. Якщо вони одного розміру, то результат буде одним числом, що в такому разі і видала система. Щоб дійсно здобути вектор значень sin(x)/x, треба використовувати спеціальний оператор поелементного ділення масивів – "./":

```
» sin(x)./x
   Warning: Divide by zero.
   ans = NaN  0.8415  0.4546  0.0470  –0.1892  –0.1918
```

Проте, і тут без особливостей не обійшлось. Оскільки, для x = 0 значення sin(x)/x утворює невизначеність при спробі ділення на 0, MatLab виводить відповідне попередження і символьну константу NaN.

Спеціальні символи

Охарактеризуємо спеціальні символи MatLab: **(:)** двокрапка – формування підвекторів і підматриць з векторів і матриць. Цей оператор – один з найбільш використовуваних операторів у системі MatLab. Він має такі правила для створення векторів:
j:k – те саме, що й [j, j+1, ..., k]; j:k – пустий вектор за умови j>k;
j:i:k – те саме, що й [j, j+i, j+2i ..... k];
j:i:k – пустий вектор за умов i>0 та j>k чи якщо i<0 та j<k, де 1, j, k – скалярні величини. За допомогою цього оператора можна вибирати рядки, стовпці та елементи з векторів, матриць і багатовимірних масивів:

A(:, j) – це j-тий стовпець з A; A(i, :) – це i-тий рядок з A;
A(:, :) – еквівалент двовимірного масиву;
A(j : k) – це A(j), A(j+l),...,A(k);
A(: j : k) – це A(:, j), A(:, j+l), ..., A(:, k);
A(:, :, k) – це k-та сторінка тривимірного масиву A;
A(i, j, k, :) – вектор, вибраний з чотиривимірного масиву A з елементами: A(1, j, k, 1), A(i, j, k, 2), A(i, j, k, 3) тощо;
A(:) – виводить усі елементи масиву A у стовпець;

( ) круглі дужки – застосовують, коли задають порядок виконання операцій в арифметичних виразах, зазначають послідовність аргументів функції або коли записують індекси елемента вектора або матриці;

[ ] квадратні дужки – використовують при формуванні векторів і матриць, наприклад:
[6.9 9.64 sqrt(–1)] – вектор з трьох елементів;
[6.9 9.64 i] – такий самий вектор;
[1+j 2–j 3] та [1 +j 2 –j 3] – різні вектори: перший містить три елементи, а другий – п’ять;
[11 12 13; 21 22 23] – матриця розміром 2 x 3. Крапка з комою відокремлює перший рядок від другого;
A = [ ] – зберігає пусту матрицю в A;
A(m, :) = [ ] – видаляє рядок m з матриці A;
A(:, n) = [ ] – видаляє стовпець n з матриці A.

{ } фігурні дужки – застосовують для формування комірок масивів. Наприклад, {magic(3) 6.9 'hello'} – масив комірок з трьома елементами;

. по-перше, крапка використовується для відокремлення цілої та дійсної частин десяткових чисел, наприклад: 314/100, 3.14 и .314e1 – одне й те саме число. По-друге, символ крапки (.) використовується для виокремлення полів структур, наприклад: A.(field) та A(i).field, де A – структура, означає виокремлення поля структури з ім’ям field;

.. батьківський каталог – перехід по дереву каталогів на один рівень уверх;

... продовження рядка – три або більше крапок наприкінці рядка означають його продовження;

, кома, розділювач – використовується для розмежування індексів елементів матриці й аргументів функції, а також для розмежування операторів мови MatLab. При розмежуванні операторів у рядку, кома може замінюватись на крапку з комою (**;**) з метою заборони виведення на екран результату обчислень;

; крапка з комою – використовується всередині круглих дужок для розмежування рядків матриць, а також наприкінці операторів для заборони виведення на екран результату обчислень;

% коментар – використовується для зазначення логічного кінця рядка. Текст після знака процента сприймається як коментар та ігнорується (на жаль, за винятком російськомовних коментарів, які частенько призводять до помилкових команд);

! знак оклику є ознакою введення команди операційної системи. Рядок, який іде за ним, сприймається як команда операційної системи;

= присвоєння значень в арифметичних виразах;

' одинарна лапка, апостроф – текст у таких лапках інтерпретується як вектор символів з компонентами, які є ASCII-кодами символів. Лапка усередині рядка задається двома лапками, наприклад:

```
» a = 'Hello"my friend'
   a = Hello'my friend;
```

' транспонування з комплексним сполученням – транспонування матриць, наприклад A' – транспонована матриця A. Для комплексних матриць транспонування виконується комплексним сполученням. Рядки транспонованої матриці відповідають стовпцям початкової матриці;

.' транспонування – транспонування масиву, наприклад A.' – транспонований масив A. Для комплексних масивів операція сполучення не виконується;

[.] горизонтальна конкатенація. Так, [A.B] – горизонтальна конкатенація (зчеплення) матриць A та B. A та B повинні мати однакову кількість рядків. [A B] діє аналогічно. Горизонтальна конкатенація може бути застосована для будь-якої кількості матриць у межах одних дужок: [A,B,C]. Горизонтальна і вертикальна конкатенації можуть використовуватись одночасно: [A,B:C];

[:] вертикальна конкатенація. Так, [A:B] – вертикальна конкатенація (зчеплення) матриць A та B. A та B повинні мати однакову кількість стовпців. Вертикальна конкатенація може бути застосована для будь-якої кількості матриць у межах одних дужок: [A:B:C]. Горизонтальна і вертикальна конкатенації можуть використовуватись одночасно: [A:B,C];

( ),{ } присвоювання підмасиву. Наведемо декілька прикладів:

A(i) = B – присвоює значення елементів масиву B елементам масиву A, які визначаються вектором індексів i. Масив B повинен мати таку саму розмірність, що й масив i, або може бути скаляром;

A(i,j) = B – присвоює значення масиву B елементам прямокутної підматриці A, які визначаються векторами індексів i та j. Масив B повинен мати length(i) рядків і length(j) стовпців;

A{i} = B – присвоює визначеній значенням скаляром I заданій комірці масиву A копію масиву B.

Формати числових даних у MatLab

За замовчуванням MatLab видає числові результати у нормалізованій *формі* з чотирма цифрами після десяткової крапки й однією до неї. Багатьох така форма подання не завжди влаштовує. Тому при роботі з числовими даними можна задавати різноманітні формати подання чисел. Однак у будь-якому разі усі обчислення здійснюються з подвійною точністю.

Формати чисел:

short − короткий формат, чотири знаки після крапки, наприклад:2.1386;

long − довгий формат, 15 цифр після крапки, наприклад: 2.1385765798124346);

short e −короткий формат в експоненціальній формі, 5 цифр мантиси й 3 цифри порядку: 2.1386e–003 (0.0021386);

long e − довгий формат в експоненціальній формі, 15 цифр мантиси й 3 цифри порядку: 2.1386e–003 (0.0021386);

short g, long g − універсальні формати. Наприклад, формат short g: 0.000013564 (1.3564e–005), 25.648 (2.5648e–001);

hex − подання чисел у шістнадцятковій формі;

bank − подання для грошових одиниць.

Для того, щоб змінити формат, використовується оператор format, наприклад:

```
» format short
» format long
```

Для ілюстрації різних форматів розглянемо вектор з двох елементів-чисел:

```
x = [4/3   1.2345e-6]
```

У різних форматах їхні подання будуть мати такий вигляд:

format short	1.3333	0.0000
format short e	1.3333E+000	1.2345E-006
format long	1.333333333333338	0.000001234500000
format long e	1.333333333333338E+000	1.2345000000000000E-006
format bank	1.33	0.00

Зазначення формату впливає лише на форми виведення чисел. Обчислення насправді здійснюються у форматі подвійної точності, а введення чисел можливе у будь-якому зручному для користувача вигляді.

Інтервальний тип даних (вектори)

Інтервальний тип даних у MatLab схожий на такий самий тип даних у Mathcad, але відрізняється синтаксисом:

Ім'я_змінної = початкове_значення **:** *крок* **:** *кінцеве_значення*

Наприклад, створимо у MatLab вектор x зі значеннями від 1 до 4 з кроком 0,5:

```
»x = 1 : 0.5 : 4
```

Після виконання цієї команди здобудемо вектор x = [1 1.5 2 2.5 3 3.5 4].

Зауважимо, що якщо крок змінення інтервальної змінної дорівнює 1, то його можна не зазначати, наприклад команда

```
»n = 1 : 10
```

створить вектор n = [1 2 3 4 5 6 7 8 9 10].

4.3 Основи редагування і створення *m*-файлів

MatLab дозволяє записувати послідовність дій програми в окремих *m*-файлах. Підготовлений і записаний на диску *m*-файл стає частиною системи, і його можна викликати як з командного рядка, так і з іншого *m*-файла. При створенні *m*-файли проходять синтаксичний контроль за допомогою вбудованого у систему MatLab редактора / налагоджувача *m*-файлів.

Існує два різновиди *m*-файлів: файли-функції та файли-сценарії:

- *файли-функції*, що мають вхідні параметри, список яких зазначається у круглих дужках. Застосовувані у файлі-функції змінні є локальними змінними;
- *файл-сценарій* або Script-файл є простим записом команд вхідних параметрів.

Для запуску Script-файла на виконання з командного рядка MatLab достатньо записати його ім’я в цьому рядку:

>> ім’я_файла.m

Для того щоб створити файл слід виконати команди з меню *File / New*, після чого відкриється вікно створення й редагування текстів програм, в якому послідовно записують і виконують програмні команди (рис. 4.2).

Редактор/налагоджувач *m*-файлів виконує синтаксичний контроль програмного коду по мірі введення тексту. При цьому використовуються такі кольорові виокремлення:

- ключові слова мови програмування – синій колір;
- оператори, константи і змінні – чорний колір;
- коментарі після знаку % – зелений колір;
- символьні змінні (в апострофах) – зелений колір;
- синтаксичні помилки – червоний колір.

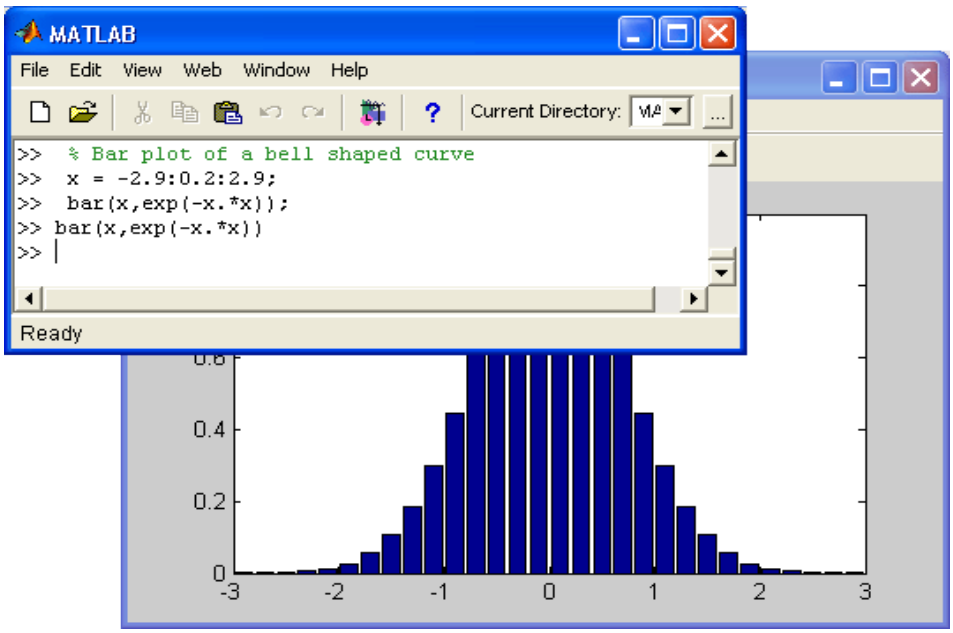


Рисунок 4.2 − Приклад виконання файла-сценарія
з командного рядка

Завдяки кольоровим виокремленням імовірність синтаксичних помилок знижується. Однак далеко не всі помилки діагностуються. Помилки, пов'язані з неправильним застосування операторів або функцій (наприклад, застосування оператора "–" замість "+" або функції cos(x) замість sin(x) тощо), не здатна виявити жодна система програмування. Усунення такого роду помилок (їх називають семантичними) – справа користувача, який налагоджує свої алгоритми і програми.

4.4 Візуалізація і графічні засоби

Останнім часом творці математичних систем приділяють величезну увагу візуалізації розв'язування математичних задач. Простіше кажучи, це означає, що постановка й опис розв'язуваної задачі та результати розв'язання мають бути ґранично зрозумілими не лише тим, хто розв'язує задачі, а й тим, хто надалі їх вивчає або просто переглядає. Значну роль у візуалізації розв'язання математичних задач відіграє графічне подання результатів, причому як кінцевих, так і проміжних. Візуалізація результатів обчислень досягається застосуванням потужних засобів графіки, у тому числі анімаційної, а також використанням засобів символьної математики.

Графічні засоби Handle Graphics (дескрипторна або описова графіка) дозволяють створювати повноцінні об'єкти графіки високого дозволу, як геометричного, так і колірного. Можливості цієї графіки підтримуються засобами об'єктно-орієнтованого програмування системи MatLab. Реалізується, до того ж з підвищеною швидкістю, побудова графіків практично усіх відомих у науці й техніці типів. Графіки виводяться окремо від текстів в окремих вікнах. На одному графіку можна вивести декілька кривих, які відрізняються кольорами і виглядом маркерів (кружки, хрестики, прямокутники тощо). Графіки можна виводити до одного чи декількох вікон.

Побудова графіків відрізками прямих

Функції однієї змінної y(x) знаходять широке застосування у практиці математичних та інших розрахунків, а також у комп'ютерному математичному моделюванні. Для відображення таких функцій використовуються графіки у декартовій (прямокутній) системі координат. При цьому, зазвичай, будуються дві осі – горизонтальна X та вертикальна Y – і задаються координати x та y, які визначають вузлові точки функції y(x). Ці точки з'єднуються одна з одною відрізками прямих, тобто при будові графіка здійснюється лінійна інтерполяція для проміжних точок. Оскільки MatLab – матрична система, сукупність точок y(x) задається векторами X та Y однакового розміру.

Команда plot призначена для створення графіків функцій у декартовій системі координат. Розглянемо можливі параметри цієї команди на прикладах.

1) plot(X, Y) – будує графік функції y(x), координати точок (x, y) беруться з векторів однакового розміру Y та X. Якщо X або Y – матриця, то будується сімейство графіків за даними колонок матриці. Наведений нижче приклад ілюструє побудову графіків двох функцій – sin(x) та cos(x), значення яких містяться у матриці Y, а значення аргументу x зберігаються у векторі X:

```
» x = [0 1 2 3 4 5];
» Y = [sin(x)  cos(x)];
» plot(x, Y)
```

На рис. 4.3 наведено графік функцій з цього прикладу. На ньому чітко видно, що графіки складаються з відрізків.

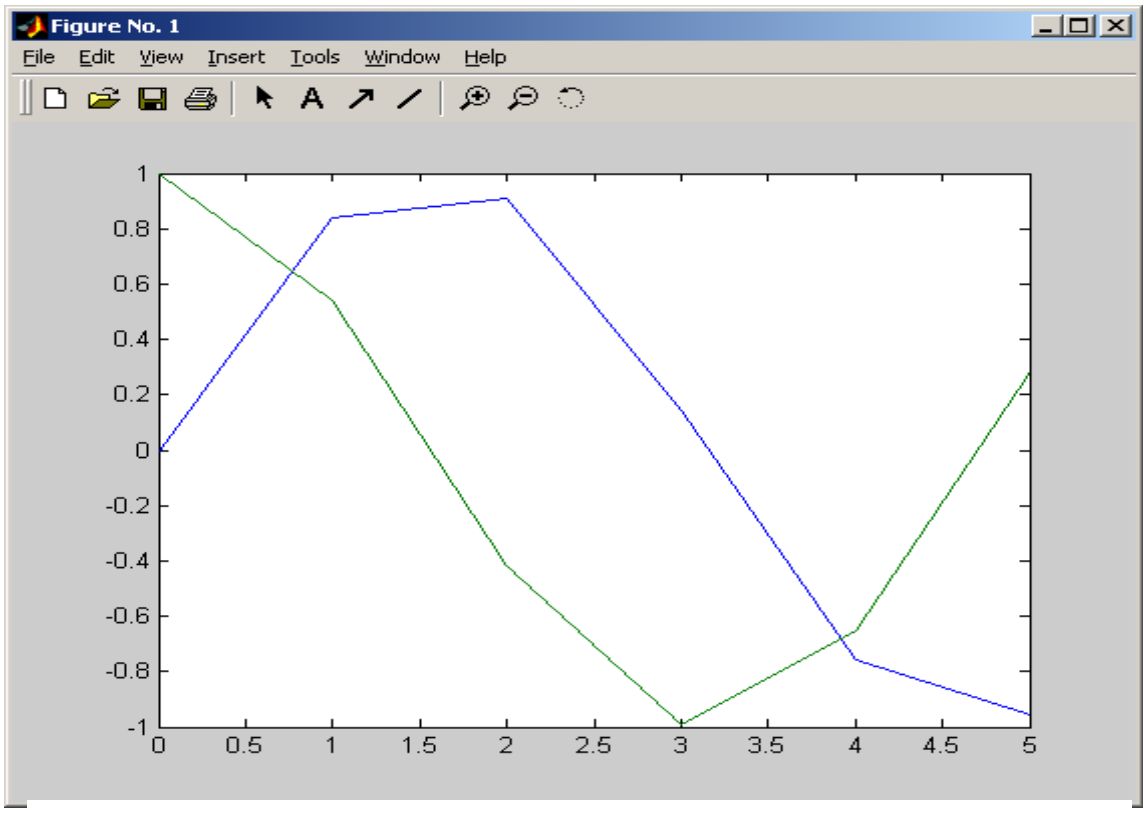


Рисунок 4.3 − Графіки двох функцій у декартовой системі координат

Якщо є потреба у відображенні функції у вигляді гладкої кривої, слід збільшити кількість вузлових точок, наприклад, задати x = 0 : 0.01 : 5 (рис. 4.4).

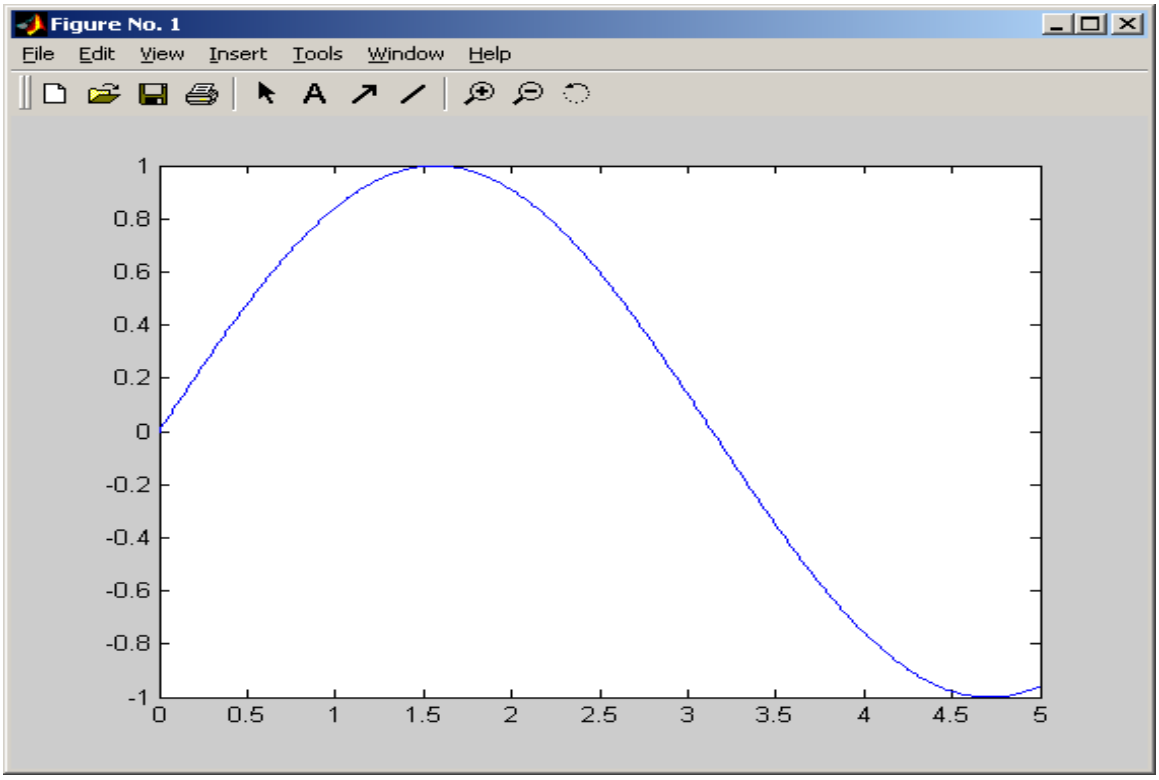


Рисунок 4.4 − Графік неперервної функції

2) plot(y) – будує графік y(n), де значення аргументу n це індекси елементів вектора y, а вісь абсцис являє собою n. Якщо y містить комплексні елементи, то виконується команда plot(real(y), imag(y)). У решті випадків уявна частина даних ігнорується. Приклад використання команди plot(y):

```
» x = 2*pi : 0.02*pi : 2*pi;
» y = sin(x) + i*cos(3*x);
» plot(y)
```

Відповідний графік наведено на рис. 4.5.

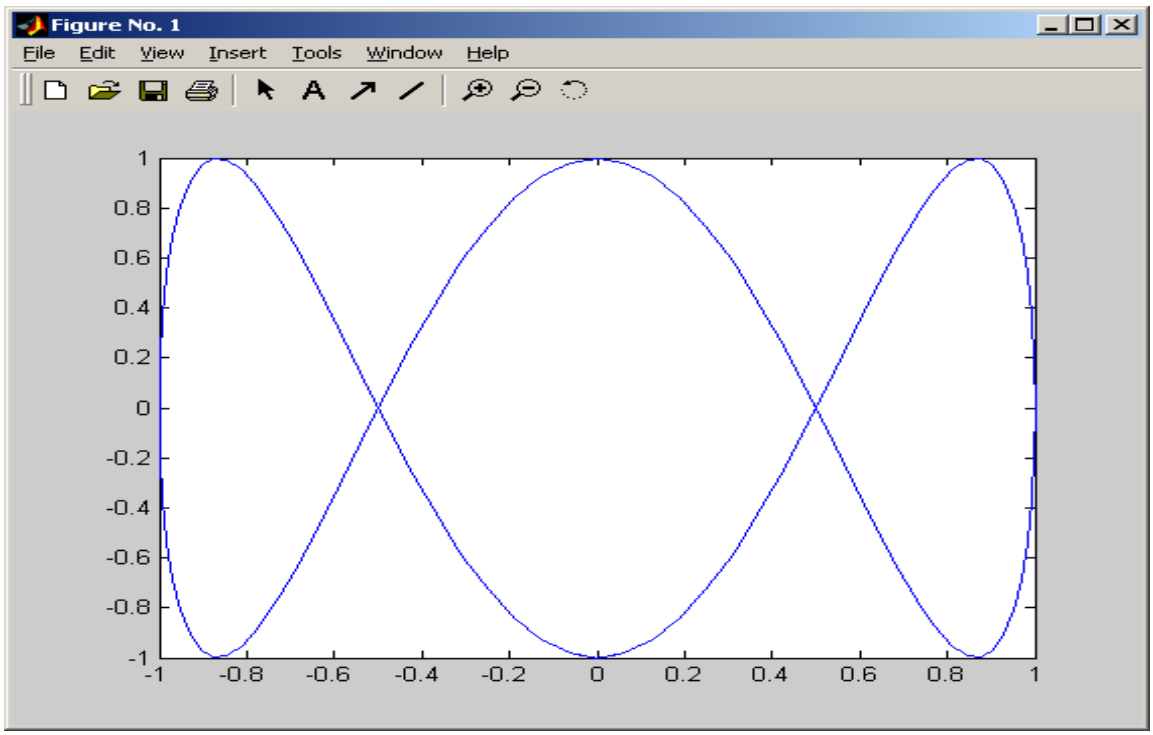


Рисунок 4.5 − Графік функції, що зображує вектор y з комплексними елементами

3) plot(X, Y, S) – є аналогічна до команди plot(X, Y), проте додатково задається тип лінії графіка значенням константи S (рядок символів):

Символ	Колір лінії
y	жовтий
m	фіолетовий
c	блакитний
r	червоний
g	зелений
b	синій
w	білий
k	чорний

Символ	Тип маркера
.	точка ●
o	кружок ○
x	хрестик ×
+	плюс +
*	зірочка ✱
s	квадрат ■
D	ромб ◆
V	трикутник ▼
A	трикутник ▲
<	трикутник ◀
>	трикутник ▶
P	п’ятикутник ⬟
H	шестикутник ⬢

Символ	Тип лінії
-	суцільна
;	подвійний пунктир
-.	пунктирна
--	штрихова

Отже, за допомогою набору з певних символів третім параметром команди plot можна змінювати тип і колір лінії, задавати вигляд маркера для вузлових точок.

4) plot (X1, Y1, S1, X2, Y2, S2, X3, Y3, S3, ...) – ця команда будує на одному графіку декілька ліній, поданих даними вигляду (X, Y, S), де X та Y – вектори або матриці, а S – рядок певних символів. За допомогою такої

конструкції можна побудувати графік функції лінією, колір якої відрізняється від кольору маркерів. Наприклад, щоб побудувати графік функції лінією синього кольору з червоними маркерами, спочатку треба побудувати графік з маркерів червоного кольору (без лінії), а потім – графік лінії синього кольору (без маркерів). За відсутності вказівок на кольори ліній та маркерів, вони обираються автоматично з таблиці кольорів (за винятком білого). Якщо ліній на одному графіку є більше шести, то кольори будуть повторюватись. Для монохромних систем лінії задають різними типами. Розглянемо приклад побудови графіків трьох функцій з різним стилем подання кожної з них (рис. 4.4):

```
» x = –2*pi : 0.1*pi : 2*pi;
» y1 = sin(x);
» y2 = sin(x).^2;
» y3 = sin(x).^3;
» plot(x, y1, '-m', x, y2, '-.+r', x, y3, '--ok')
```

Як видно з рисунка, графік функції y1 будується суцільною фіолетовою лінією, графік y2 будується пунктирною лінією із маркерами у вигляді знака “плюс” червоного кольору, а графік y3 будується штриховою лінією з кружками чорного кольору. Слід враховувати, що при роздруковуванні на чорно-білих принтерах на папері кольорові лінії можуть зображуватись дуже схожими між собою градаціями сірих кольорів.

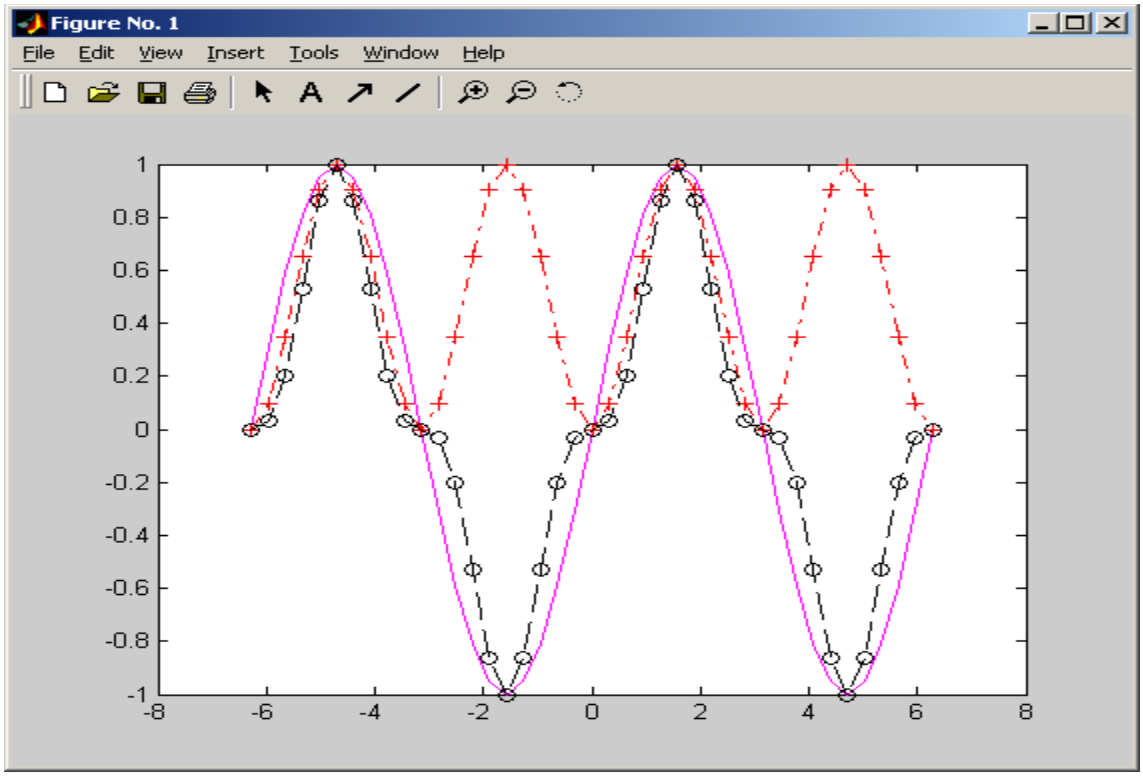


Рисунок 4.6 − Вигляд трьох функцій на одному графіку з різними стилями ліній

4.5 Статистичні функції MatLab

1) sum(x) – обчислює суму елементів стовпців матриці, (якщо x – вектор, обчислюється сума елементів вектора $\sum_{i=1}^{n} x_i$). Наприклад, задамо значення матриці A та обчислимо суми елементів рядків S і суми елементів стовпців S1:

```
»A = [ 1 2 3;  4 5 6;  7 8 9 ]
   A=[1  2  3
      4  5  6
      7  8  9]
»S = sum(A)       %  вектор сум елементів стовпців
      S = [12  15  18]
»S1= sum(A')      %  вектор сум елементів рядків
      S1 = [6  15  24]
```

2) mean(x) – обчислює середні значення елементів стовпців матриці, а якщо x – вектор з *n* елементів, то обчислюється $m_x = \frac{1}{n}\sum_{i=1}^{n} x_i$. Наприклад:

```
» MA=mean(A)
      MA = [4  5  6]
» MS=mean(S)
 S= 15
```

3) min(x) – обчислює мінімальне значення вектора або елементів стовпців матриці, наприклад для матриці A :

```
» Mn= min (A)
Mn = [1  2  3]
```

4) max(x) – максимальне значення вектора або елементів стовпців матриці, наприклад для матриці A:

```
» Mx= max(A)
Mx= [7  8  9]
```

5) std(x) – стандартні (середньоквадратичні) відхилення значень елементів стовпців матриці. Якщо x – вектор з *n* елементів, то стандартне відхилення (δ_s) та середньоквадратичне (δ_k) відхилення значень елементів стовпців обчислюються за формулами:

$$\delta_s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - m_x)^2}\ ,\ \delta_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)^2}\ .$$

Наприклад, обчислимо стандартне відхилення (δ_s) для вектора S:

```
» Ssv= std (S)
   Ssv = 3
```

Зауважимо, що для обчислення середньоквадратичного відхилення (δ_s) функцію std записують з додатковим параметром (1), наприклад:

```
» Ssv= std (S,1)
    Ssv=2.4495
```

6) sort(y) – функція *сортування за зростанням*. Наприклад:

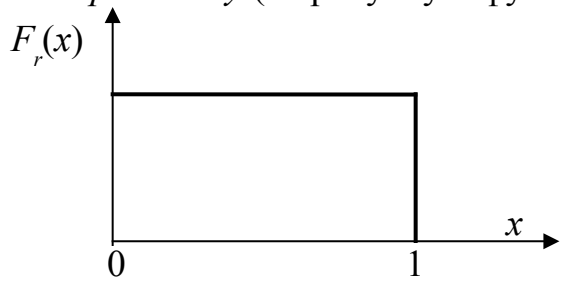
```
»y =  [3 1 8 4 5];
»w = sort(y)
        w = [1  3  4  5  8]
```

Для *сортування за спаданням* запишемо:

```
»b = − sort(-y)
        b = [8  5  4  3  1]
```

4.6 Формування випадкових чисел

1) x = rand(n) формує матрицю x розміром $n \times n$ випадкових чисел від −1 до 1 з *рівномірним законом розподілу* (на рисунку − функція розподілу $F_r(x)$).



Варіанти використання цієї функції:

rand(n, m) – формує матрицю випадкових чисел з *n* рядків та *m* стовпців;

rand(n, 1) – створює стовпець з *n* випадкових чисел;

rand(1, n) – створює рядок з *n* випадкових чисел.

Можна змінювати початкове значення, з якого починається відлік випадкових чисел за допомогою функції rand('seek', k), де k – велике ціле непарне число. Наприклад, створимо стовпець зі 100 випадкових чисел із значеннями в проміжку [−1, 1] за рівномірним законом розподілу:

```
»rand('seek' ,10001); x = rand(100, 1);
```

Для створення вектора випадкових чисел y з рівномірним розподілом зі значеннями на проміжку [a, b] необхідно записати команди:

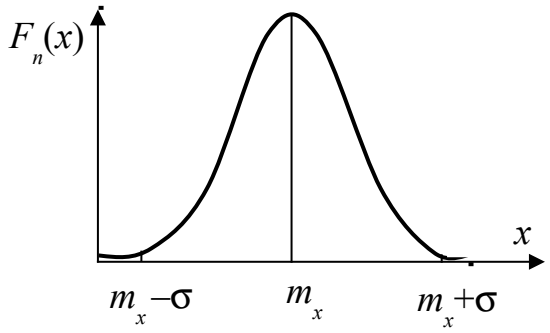
```
»x = rand(n, 1); y=a+(b-a).*x;
```

2) randperm(n) – формує масив з *випадковою перестановкою n* цілих чисел:

```
»x = randperm(6)
x = [5  31  4  2  6]
```

3) x = randn(n, m) – створює матрицю випадкових чисел за *нормальним законом розподілу* із математичним очікуванням нуль (0) і середньоквадратичним відхиленням одиниця (1). Вектор чисел за нормальним законом розподілу із заданими математичним очікуванням m_x і середньоквадратичним відхиленням σ (на рисунку − функція розподілу $F_n(x)$) обчислюється за формулою $y = m_x + \sigma * x$

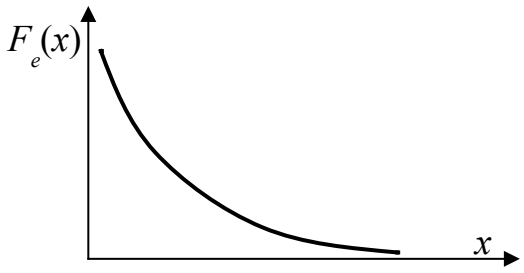
:



Для створення вектора випадкових чисел y з нормальним законом розподілу зі значеннями $m_x = mx$ та $\sigma = sx$ необхідно записати команди:

```
»x = randn(n, 1); y=mx+sx.*x;
```

4) вектор випадкових чисел z за *експоненціальним (Пуассона) законом* розподілу (на рисунку − функція експоненціального розподілу $F_e(x)$) із заданим середнім значенням m_x обчислюється за формулою $z = -m_x \ln(1 - x)$:



де x − вектор випадкових чисел зі значеннями в проміжку [−1, 1] за рівномірним законом розподілу. Команди формування вектора за експоненціальним законом розподілу із значенням $m_x = mx$:

```
»x = rand(n, 1); z = –mx.* ln(1–x);
```

ТЕМА 5. ПОСТАНОВКА ЗАДАЧІ ОПТИМІЗАЦІЇ. ЧИСЕЛЬНІ МЕТОДИ ПОШУКУ ЕКСТРЕМУМУ ФУНКЦІЇ З ОДНІЄЮ ЗМІННОЮ: РІВНОМІРНИЙ, БІСЕКЦІЙ, ЗОЛОТОГО ПЕРЕТИНУ

5.1 Постановка математичної задачі оптимізації

У достатньо загальному вигляді математичну задачу оптимізації можна сформулювати так: мінімізувати (максимізувати) цільову функцію з урахуванням обмежень на керуючі змінні. Тобто для заданої функції $F(x)$ треба знайти $\bar{x}$, яке є мінімумом (або максимумом) цієї функції на інтервалі $[a, b]$ із заданою точністю ε, тобто знайти

$$\bar{x} = \min\{F(x)\}, \quad \bar{x} \in [a, b]$$

або

$$\bar{x} = \max\{F(x)\}, \quad \bar{x} \in [a, b].$$

Функція $F(x)$ може мати кілька екстремальних (мінімальних і максимальних) значень.

Одновимірна оптимізація (пошук екстремумів функцій однієї змінної) є найпростішою і поширеною математичною задачею оптимізації, в якій цільова функція залежить від однієї змінної. Крім того, до неї зводиться набагато більш складна задача – пошук екстремуму функції багатьох змінних.

5.2 Метод рівномірного пошуку екстремуму

Метод рівномірного пошуку мінімального значення функції заснований на тому, що змінній x присвоюється значення $x + \Delta x$ із кроком $\Delta x = \text{const}$ і обчислюються значення $F(x)$. Якщо $F(x_{n+1}) > F(x_n)$, змінній x надається новий приріст. Як тільки-но $F(x_{n+1})$ стане менше за $F(x_n)$, пошук припиняється (рис. 5.1). При малій заданій похибці метод є неекономічний за витратами машинного часу. Схему алгоритму методу наведено на рис. 5.2.

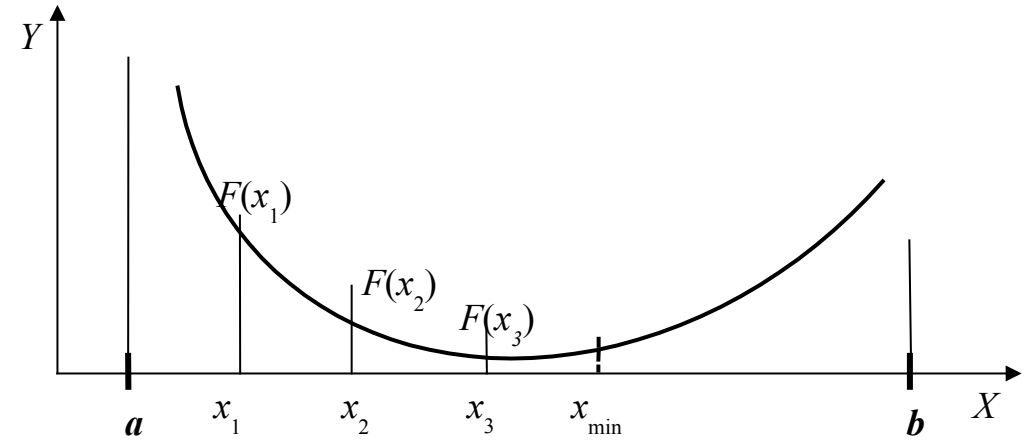


Рисунок 5.1 – Оптимизация методом рівномірного пошуку

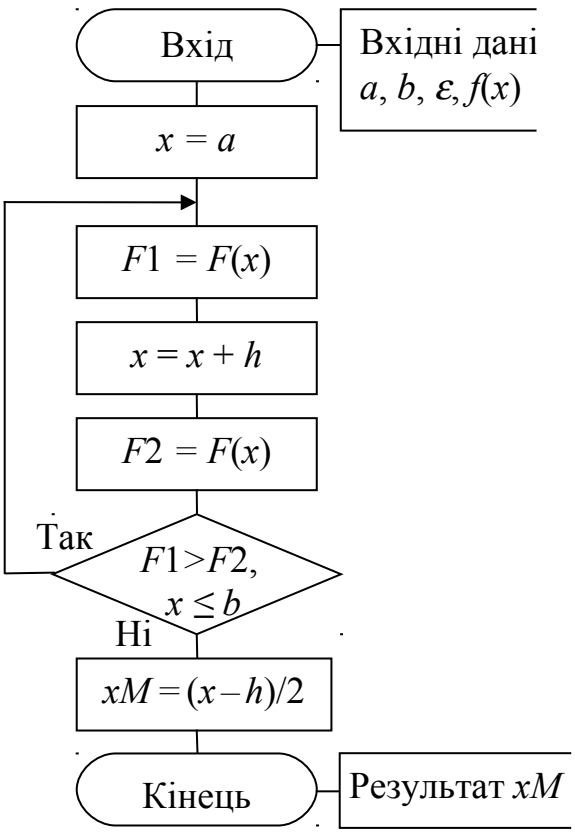


Рисунок 5.2 – Блок-схема методу рівномірного пошуку екстремуму

5.3 Метод бісекції

Метод бісекції, відомий також під назвами *метод ділення навпіл* або *метод дихотомії*, – найпростіший, надійний, але порівняно повільний метод. Суть методу в тому, що інтервал ділиться навпіл, обраховується значення функції в середній точці, й порівнюється її знак зі знаками функції на кінцях інтервалу. Така процедура дозволяє виділити наполовину менший інтервал із різними знаками функції на його кінцях. Її повторяють доти, доки довжина інтервалу не стане меншою від заданої точності.

Цей метод є *методом прямого пошуку*. У ньому при пошуку екстремуму цільової функції використовують тільки обчислені значення цільової функції.

Запишемо докладний словесний алгоритм методу:

1). На кожному кроці процесу пошуку поділити відрізок [*a, b*] навпіл, тобто обчислити координату середини відрізка [*a, b*] за формулою

$$x = (a + b)/2.$$

2). Обчислити значення функції $F(x)$ в околі $\pm\varepsilon$ обчисленої точки x (рис. 5.3):

$$F1 = F(x - \varepsilon),$$

$$F2 = F(x + \varepsilon).$$

3). Порівняти $F1$ та $F2$ і відкинути одну з половинок відрізка $[a, b]$ (рис. 5.3):

а) *при пошуку мінімуму методом бісекції*: якщо $F1 < F2$, відкинути відрізок $[x, b]$, тоді $b = x$ (рис. 5.3, а), інакше – відкинути відрізок $[a, x]$, тоді $a = x$ (рис. 5.3, б);

б) *при пошуку максимуму*: якщо $F1 < F2$, відкинути відрізок $[a, x]$, тоді $a = x$, інакше – відкинути відрізок $[x, b]$, тоді $b = x$.

4). Ділення відрізка $[a, b]$ триває, доти поки його довжина не стане меншою за задану точність ε, тобто $|b - a| \le \varepsilon$.

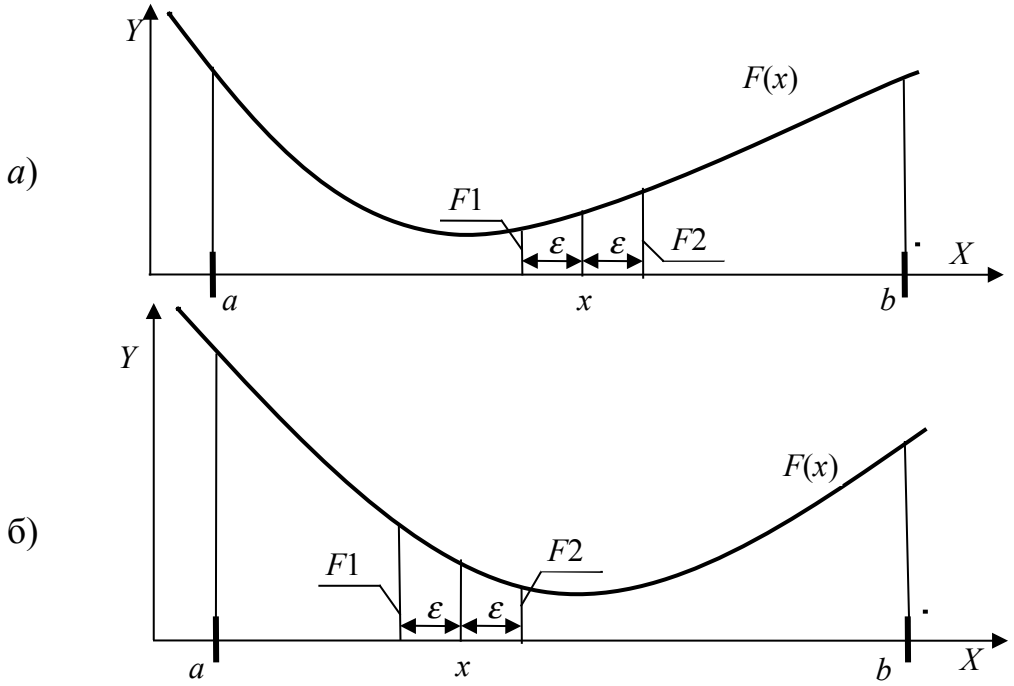


Рисунок 5.3 – Пошук екстремуму функції $F(x)$ методом бісекції (дихотомії)

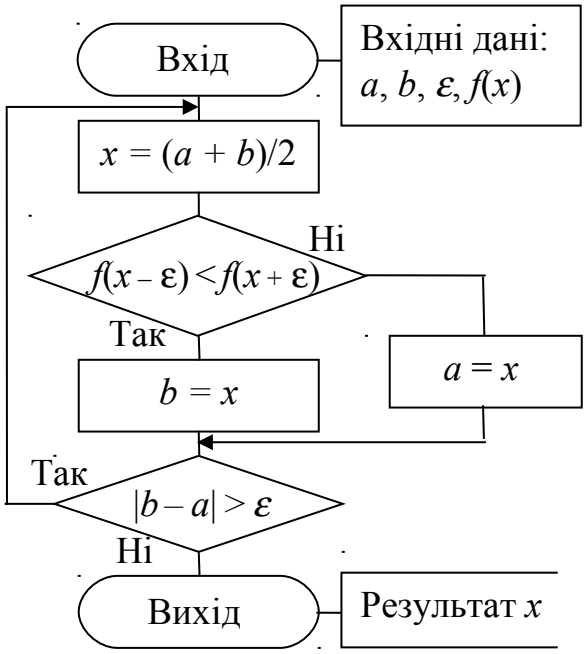


Рисунок 5.4 − Блок-схема функції пошуку екстремуму методом ділення навпіл

Схему алгоритму *методу бісекції* подано на рис. 5.4. У блоці виведення результатів x – координата точки, у якій функція $F(x)$ має мінімум (або максимум), FM – значення функції $F(x)$ у цій точці.

5.4 Метод "золотого перетину"

Розглянемо найбільш поширений метод одновимірної оптимізації – *метод "золотого перетину"*.

Точка "золотого перетину" x_1 поділяє відрізок $[a, b]$ на дві частини так, що відношення більшої частини відрізка до цілого відрізка дорівнює відношенню меншої частини до більшого, тобто дорівнює так званому "золотому відношенню" (рис. 5.5):

$$\frac{b - x_1}{b - a} = \frac{x_1 - a}{b - x_1} = \frac{\sqrt{5} - 1}{2} \approx 0{,}618. \qquad (1)$$

На рис. 5.5 проілюстровано "золоте відношення" відрізків, де відрізки, позначені дугами догори, відповідають першому дробу рівняння (1), а відрізки, позначені дугами донизу, відповідають другому дробу рівняння. Точка x_2 є правою симетричною точкою "золотого перетину" на відрізку $[a, b]$, якщо для неї виконується умова:

$$\frac{x_2 - a}{b - a} = \frac{b - x_2}{x_2 - a}.$$

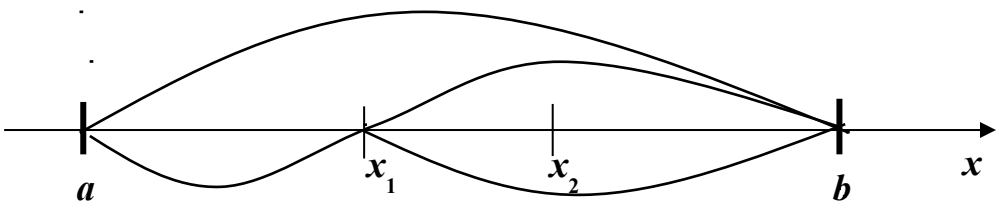


Рисунок 5.5- Відношення відрізків за методом "золотого перетину"

Алгоритм пошуку екстремуму методом "золотого перетину" (див. рис.5.6).

1) Обчислюємо значення симетричних точок "золотого перетину"

$x_1 = a + (1 - \tau)(b - a)$, $x_2 = a + \tau(b - a)$, де $\tau = 0.618$ коефіцієнт "золотого відношення".

2) Обчислюємо значення функції в точках "золотого перетину"

$$F1 = F(x_1), \qquad F2 = F(x_2).$$

3) Порівнюємо $F1$ і $F2$ і відкидаємо одну з половинок відрізка $[a, b]$.

При пошуку мінімуму:

якщо $F1 < F2$, то відкидаємо відрізок $[x_2, b]$, тоді $b = x_2$, інакше відкидаємо відрізок $[a, x_1]$, тоді $a = x_1$ (див. рис. 5.6).

При пошуку максимуму:

якщо $F1 < F2$, то відкидаємо відрізок $[a, x_1]$, тоді $a = x_1$, інакше відкидаємо відрізок $[x_2, b]$, тоді $b = x_2$.

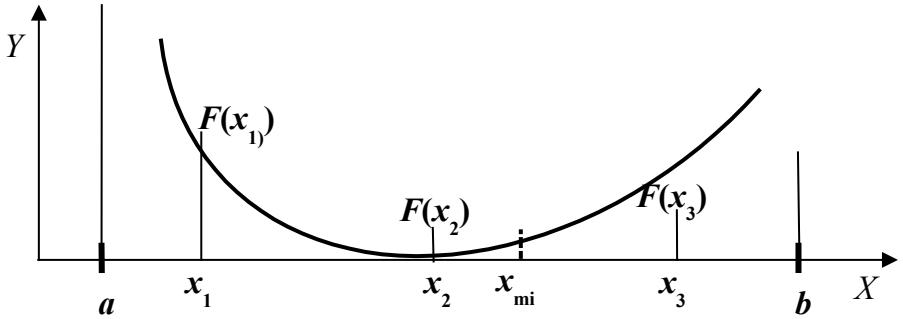


Рисунок 5.6 – Оптимізація методом “золотого перетину”

4) Для знайденого у п. 3 зменшеного відрізка обчислюємо симетричну точку “золотого перетину” (наприклад, $x_3 = x_1 + (1 - \tau)\,(b - x_1)$ – праву точку “золотого перетину” відрізка $[x_1, b]$) і обчислюємо значення функції $F(x_3)$. Якщо ввести позначення $F1 = F2$, $F2 = F(x_3)$, то повернувшись до п. 3, можна продовжувати пошук екстремуму.

5) Ділення відрізка $[a, b]$ триває, доти поки його довжина не стане меншою за задану точність ε, тобто $|b - a| \le \varepsilon$.

Блок-схему алгоритму подано на рис. 5.7. На блок-схемі c – це константа:

$$c = \begin{cases} 1 & (\text{пошук мінімуму функції } F(x)); \\ -1 & (\text{пошук максимуму функції } F(x)). \end{cases}$$

При виведенні x – координата точки, у якій функція $F(x)$ має мінімум (або максимум), FM – значення функції $F(x)$ у цій точці.

Метод гарантує знаходження мінімуму у най несприятливіших умовах і вимагає удвічі менше обчислень значень функції $F(x)$ порівняно з методом бісекції, однак він має більш повільну збіжність, оскільки на кожному кроці довжина відрізка для пошуку мінімуму зменшується тільки в 1,7 рази.

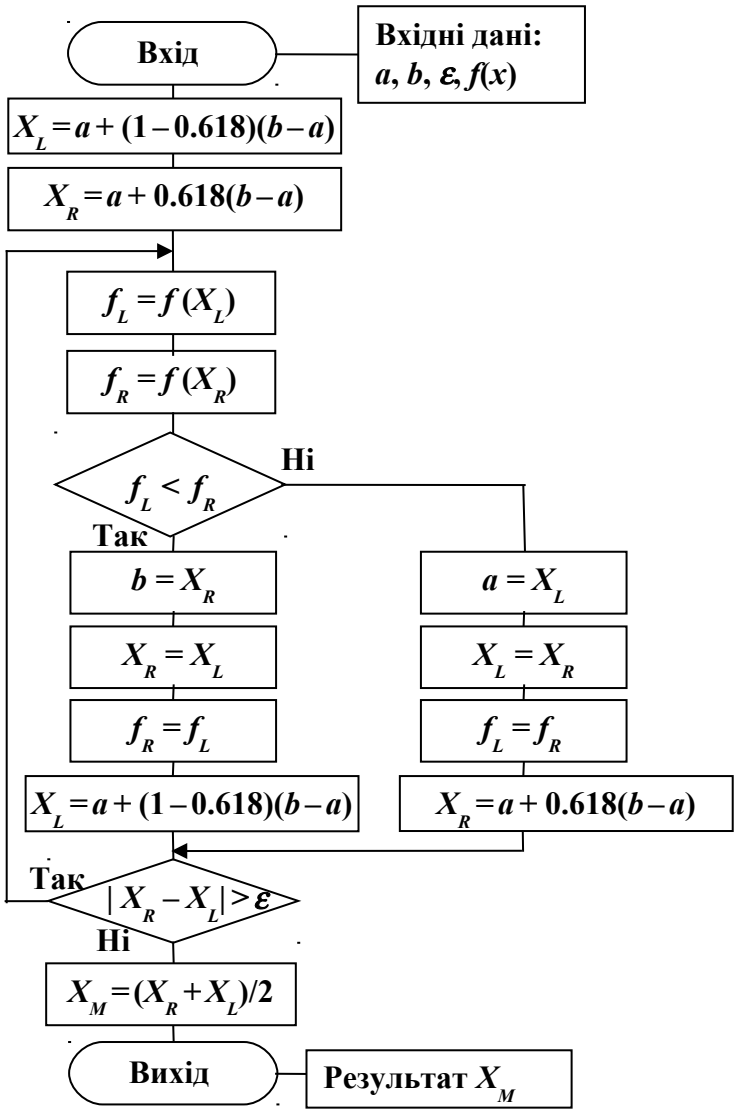


Рисунок 5.7 – Схема алгоритму методу “золотого перетину”

ТЕМА 6. ОПТИМІЗАЦІЯ БАГАТОВИМІРНИХ ФУНКЦІЙ. ЧИСЕЛЬНІ МЕТОДИ ОПТИМІЗАЦІЇ БАГАТОВИМІРНИХ ФУНКЦІЙ: МЕТОД КООРДИНАТНОГО СПУСКУ, МЕТОД ГРАДІЄНТНОГО СПУСКУ

6.1 Постановка задачі

На перший погляд може здатися, що відмінність між методами багатовимірного і одновимірного пошуку полягає лише в тому, що перші вимагають більшого обсягу обчислень, і, що методи, які придатні для функцій однієї змінної, можна застосовувати і для функцій багатьох змінних. Але це не так, оскільки багатовимірний простір якісно відрізняється від одновимірного. Передусім зі збільшенням числа вимірів зменшується ймовірність унімодальності функції. Крім того, безліч елементів, які утворюють багатовимірний простір, значно потужніші множини елементів одновимірного простору. Обсяг обчислень, які необхідні для звуження інтервалу невизначеності в багатовимірному просторі, є степеневою функцією, показник якої дорівнює розмірності простору. Так, якщо у випадку одновимірного простору для досягнення точності 0,1 треба обчислити 19 значень цільової функції, то у випадку двовимірного простору це число складає 361, тривимірного – 6859, чотиривимірного – 130321, а п'ятивимірного – 2476099! Оскільки при виборі оптимальної конструкції нерідко потрібно мати справу з п'ятьма і більше змінними, серйозність труднощів, зумовлених багатовимірністю, стає очевидною.

Оптимізацію функції багатьох змінних розглянемо на прикладі функції двох змінних *F(x,y),* яка описує деяку поверхню в тривимірному просторі з координатами *x,y.* Завдання $F(x,y) = \min$ означає пошук нижчої точки цієї перерхні. Тобто нам потрібно знайти таку точку (x_m, y_m), для якої при деякому достатньо малому числі $\varepsilon>0$ для всіх точок, що не збігаються із згаданою точкою і задовольняють умові $|x-x_m|\le\varepsilon$, $|y-y_m|\le\varepsilon$, виконується нерівність $F(x,y)\ge F(x_m,y_m)$. Необхідна умова існування мінімуму (максимуму) у точці (x_m,y_m) є $\partial F / \partial x = \partial F / \partial y = 0$. Достатня умова екстремуму в цій точці є необхідність, щоб усі другі похідні функції були позитивні (від'ємні). Як в топографії, зобразимо рельєф цієї поверхні лініями рівня. За видом ліній рівня умовно виділяють три типи рельєфу: улоговинний (рис. 6.1), яружний (рис. 6.2) і неврегульований (рис. 6.3). Два останні типи рельєфу шляхом символьних перетворень можна перевести до улоговинного типу. При улоговиному рельєфі лінії рівнів схожі на еліпси. В малій околиці невиродженого мінімуму рельєф функції є улоговий. Це видно з наступних міркувань. У мінімуму перші похідні функції $F(x_m,y_m)$ дорівнюють нулю $\partial F / \partial x = \partial F / \partial y = 0$, розкладання функції в ряд Тейлора поблизу мінімуму має вигляд :

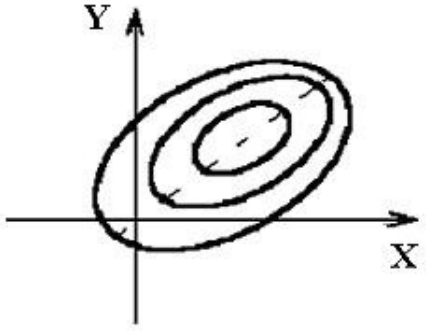


Рисунок 6.1 Улоговий рельєф

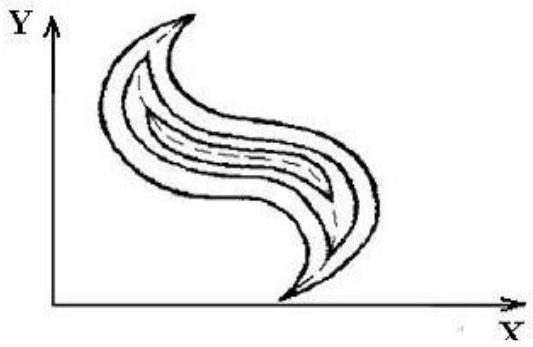


Рисунок 6.2 − Яружний рельєф

.

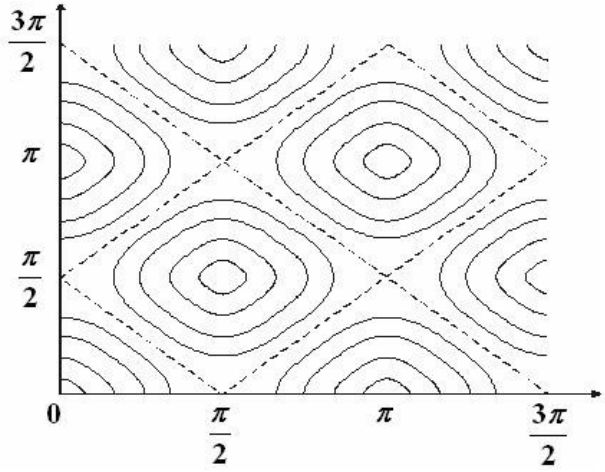


Рисунок 6.3 – Невпорядкований рельєф

.

$$F(x, y) = F(x_m, y_m) + 1/2\,(\Delta x)^2 F_{xx} + + \Delta x \Delta y F_{xy} + 1/2\,(\Delta y)^2 F_{yy} + \ldots . \qquad (6.1)$$

де F_{xx}, F_{yy} − друга похідна за x та y, F_{xy} – похідна за x та y, $\Delta x = x - x_m$, $\Delta y = y - y_m$.
Тому що квадратична форма – позитивно визначена, а лінії рівня знакозмінної квадратичної форми - це еліпси. Яружний тип рельєфу, це коли лінії рівня кусково-гладкі, тобто мають точки зламу. Геометричне місце точок зламу. називається істинним яром, якщо кут направлений в сторону зростання функції, і гребенем – якщо в сторону зменшення. Найчастіше лінії рівня всюди гладкі, але на них є ділянки з значною кривизною - розв'язні яри або гребені. Наприклад, рельєф функції

$$F(x, y) = 10\,(y - \sin x)^2 + 0{,}1\,x^2 . \qquad (6.2)$$

У фізичних задачах яружних рельєф вказує на те, що не врахована якась закономірність, наявна між змінними. Наприклад, якщо для функції (6.2) ввести нові змінні $t = x$, $l = y - \sin x$, то рельєф стане улоговий. Невпорядкований тип рельєфу характеризується наявністю багатьох максимумів, мінімумів і сідловин. Наприклад:

$$F(x,y) = (1+\sin 2x)(1+\sin 2y). \qquad (6.3)$$

Усі ефективні методи пошуку мінімуму багатовимірної функції зводяться до побудови траєкторії, вздовж якої функція зменшується. Різні методи відрізняються лише засобами побудови цих траєкторій. Здавалося б, для знаходження мінімуму достатньо розв’язати систему рівнянь типу (6.1) і відкинути ті розв’язання, які є сідловинами або максимумами. Проте в реальних задачах мінімізації методи розв’язання таких систем звичано сходяться у настільки малій околиці мінімуму, що вибрати відповідне нульове наближення не завжди вдається.

6.2 Метод координатного спуску

Простіше й ефективніше провести спуск за координатами. Викладемо цей метод на прикладі функції двох змінних $F(x, y)$. Сутність методу при відшукуванні, наприклад, мінімуму полягає в тому, що кожну з незалежних змінних змінюють доти, поки досліджувана функція – поверхня $F(x, y)$ не

перестане спадати, потім переходять до іншої змінної. Після завершення мінімізації функції у першому циклі переходять до другого циклу.

Виберемо нульове наближення x_0, y_0. Фіксуємо значення однієї координати $y = y_0$, а іншу відпускаємо, так отримаємо функцію $f1(x)$. Знайдемо мінімум цієї функції будь-яким методом мінімізації функції однієї змінної та позначимо його x_1. Ми зробили крок з точки (x_0, y_0) в точку (x_1, y_0) за напрямом осі x. Потім з нової точки зробимо спуск у напрямку осі y, тобто розглянемо $f2\ (y) = F(x_1, y)$, знайдемо її мінімум і позначимо його y_1. Другий крок приводить нас в точку (x_1, y_1), тим самим ми завершили перший цикл спусків. Будемо повторювати цикли. На кожному спуску функція не зростає, і при цьому значення функції обмежені знизу її значеннями в мінімумі Fm. Отже, ітерації будуть сходитися до деякої межі. Якість методу координатного спуску суттєво залежить від виду ліній рівня. Оскільки кожна ланка траєкторії пошуку паралельна одній з координатних осей, і загальна траєкторія руху в області визначення утворює «сходи», метод є найбільш ефективний у тому випадку, коли лінії рівня являють собою кола або еліпси. Проте у випадку справжнього яру спуск по координатах приведе нас на дно яру тільки по одній координаті, а будь-який рух за такою координаті веде нас на підйом. Ніякий подальший спуск по координатам неможливий, хоча мінімума ще не досягнуто.

6.3 Метод градієнтного спуску

Відомо, що *напрямок градієнта* є напрямком найшвидшого зростання функції. Отже, протилежний напрямок є напрямком найшвидшого зменшення функції. Градієнтом у математиці для функції $f(x_0,x_1,\ldots,x_n)$ називається вектор з координатами $\left\{\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right\}$, який позначається *grad f* або ∇f. Тому, якщо ми знаходимося в точці x_i на деякому кроці процесу оптимізації, то пошук мінімуму функції здійснюється вздовж напрямку $-\nabla f(x_i)$. Даний метод є ітераційним. На кроці i точка мінімуму апроксимується точкою x_{i-1}. Наступною апроксимацією є

$$x_{i+1} = x_i - \lambda_i \nabla f(x_i) \qquad (6.4)$$

де λ_i – значення λ, мінімізує функцію

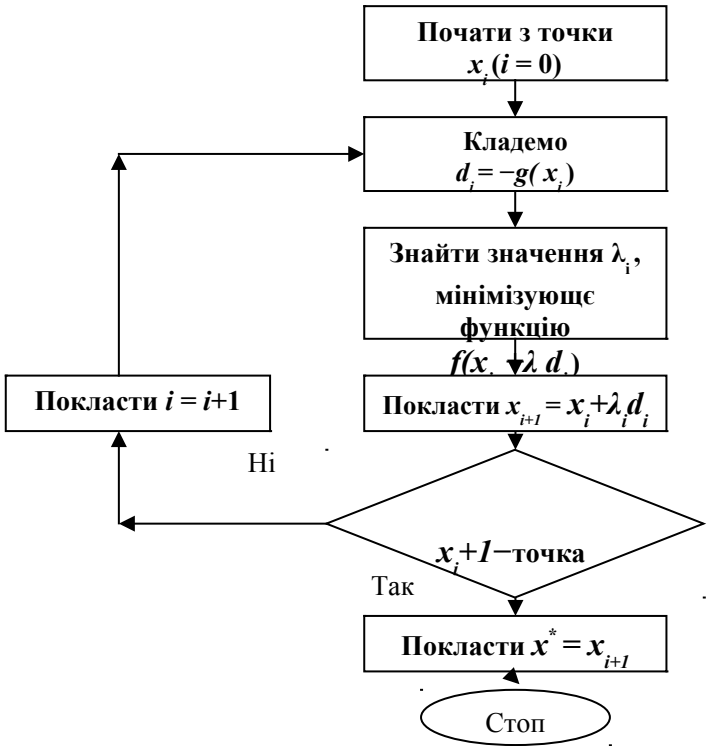


Рисунок 6.4 – Блок-схема методу градієнтного спуску

$$\varphi(\lambda) = f[x_i - \lambda \nabla f(x_i)]. \qquad (6.5)$$

Значення λ_i може бути знайдено за допомогою одного з методів одновимірного пошуку. У цій точці знову визначається градієнт і робиться наступний спуск. Цей метод складніше метода координатного спуску, тому що вимагає обчислення похідних і градієнта та переходу до інших змінних. Блок-схема методу градієнтного спуску наведена на рис. 6.4.

ТЕМА 7. ОПТИМІЗАЦІЯ БАГАТОВИМІРНИХ ФУНКЦІЙ З ОБМЕЖЕННЯМИ: ЦІЛЬОВА ФУНКЦІЯ ТА ФУНКЦІЇ ОБМЕЖЕННЯ. ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ. ГРАФІЧНИЙ МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

7.1 Постановка задачі оптимізації функції з обмеженнями

Нехай в n - вимірному просторі задана функція $F(x)$. Розглянемо задачу на мінімум з додатковими умовами двох типів:

$$F(x) = min, \quad \varphi_i(x) = 0,\ 1 \le i \le m, \quad \psi_j(x) \ge 0,\ 1 \le j \le p. \tag{7.1}$$

Умови типу рівностей виділяють в просторі якусь (n-m)-вимірну поверхню, тому має виконуватися нерівність $m < n$. Умови типу нерівностей виділяють (n-m)-вимірну область R, яка обмежена гіперповерхнями $\psi_j(x) = 0$, число таких умов може бути довільним. Отже задача 7.1– є пошуком мінімуму функцій n змінних в (n-m)-мірній області R. Функція $F(x)$ у такій задачі називається ***цільовою***, а функції **φ** і **ψ** - ***функціями обмеження***.

7.2 Постановка задачі лінійного програмування

При оптимізації економічних планів виникають задачі на пошук мінімуму лінійний функції n змінних за наявності лінійних додаткових умов трьох типів:

$$F(x) = \Sigma c_i x_i = \min, \quad x_i \ge 0,\ 1 \le i \le n, \quad \Sigma a_{ji} x_i = b_j,\ 1 \le j \le m,\ \Sigma a_{ji} x_i \le b_j,$$
$$m \le j \le \text{M (сума по } i \text{ від 1 до } n). \tag{7.2}$$

Такі задачі називаються задачами лінійного програмування.

Кожна з умов типу нерівностей визначає півпростір, обмежений гіперплощиною, усі ці умови разом визначають опуклий n-вимірний багатогранник R, який є перетином відповідних півпросторів. Умови типу $\Sigma a_{ji} x_i = b_j$ виділяють з n-вимірного простору (n-m)-вимірну площину. Її перетин з областю R дає опуклий (n-m)-вимірний багатогранник G. Наша задача полягає в тому, щоб знайти мінімум лінійної функції $F(x)$ в цьому багатограннику. Багатогранник умов - опуклий, тому всередині нього лінійна функція $F(x)$ не може досягати мінімуму. Її мінімум, якщо він існує, досягається обов'язково у будь-якій вершині багатогранника. При виродженні він може досягатися в усіх точках ребра або навіть p-вимірній обмеженій площині. Тому завдання лінійного програмування теоретично просте, достатньо обчислити значення функції в кінцевому числі точок – у вершинах багатогранника і знайти серед цих значень найменше. Знаходити вершини багатогранника за дуже великого числа змінних незручно. Краще перетворити завдання до *канонічної форми*, яка не містить умов третього типу. Для цього в якості нових змінних введемо нев'язки для умов третього типу,

$$x_i = b_{i+m-n} - \sum_{q=1}^{n} a_{i+m-n,q} x_q \ge 0, \qquad n < i < N,\ N = n + M\text{-}m. \tag{7.3}$$

Довизначимо коефіцієнти цільової функції та функцій обмежень таким чином $c_i = 0$, $a_{ji} = \delta_{j,i+M-N}$ при $n < i < N$, $1 < j < M$. Тоді задача лінійного програмування набуде канонічного виду:

$$F(x) = \Sigma c_i x_i = min,\ x_i \geq 0,\ 1 \leq i \leq N,\ \Sigma a_{ji} x_i = b_j,\ 1 \leq j \leq M\ (M < N). \qquad (7.4)$$

Багатогранник нових канонічних умов є утворений перетином нової (N-M)-вимірною площини умов з першим координатним кутом, це означає, що усі його вершини лежать на координатних гіперплощинах, тобто у кожної вершини частина координат – нулі, а решта координат – позитивні. Тоді ранг матриці A утвореної з елементів a_{ji} дорівнює M і, отже серед її стовпців, знайдеться M лінійно-незалежних стовпців. Усе лінійно-незалежні набори стовпців матриці A відповідають точкам перетину площини умов з координатними гіперплощинами.

Щоб знайти вершину, візьмемо один набір таких стовпців. Перепишемо умови другого типу з (7.4) в такому вигляді:

$$\sum_{i=1}^{M} a_{ji} x_i = b_j - \sum_{i=M+1}^{N} a_{ji} x_i, \qquad 1 \leq j \leq M \qquad (7.5)$$

Позначимо через α_{ji} елементи матриці, оберненої до базисної, прирівняємо позабазисні координати нулю, розв'язуємо отриману систему, і визначаємо координати точки перетину площини умов з координатною гіперплощиною:

$$x_j = \sum_{i=1}^{M} \alpha_{ji} b_j \qquad 1 \leq j \leq M,\ x_j = 0, \text{ якщо } M < j \leq N \qquad (7.6)$$

Якщо знайдені координати позитивні, точка є вершиною багатогранника канонічних умов. Якщо хоча б одне $x_j < 0$, цю точку треба відкинути та досліджувати інший набір стовпців матриці A. У найсприятливіших умовах багатогранник умов може мати до 2^N вершин і якщо $N = 100$ це число настільки велике, то такий перебір можуть зробити тільки найпотужніші комп'ютери.

7.3 Графічний метод розв'язання задачі лінійного програмування

Для найпростішої задачі лінійного програмування (ЛП) з двома змінними розв'язання може бути виконане за допомогою графічного методу. Графічне розв'язання задачі ЛП з двома змінними та функціями обмеження у виді лінійних нерівностей складається з двох етапів.

1) Побудова на координатній площині множини розв'язань системи лінійних нерівностей, що є опуклим багатогранником.
2) Вибір у множині припустимих розв'язань точки $A(x_{1m}, x_{2m})$, в якій цільова функція має максимальне (мінімальне) значення.

Розглянемо приклад найпростішої задачі лінійного програмування:

$\max(x_1 + x_2)$ – цільова функція,

функції обмеження: $3x_1 + 5x_2 \leq 15,\quad -x_1 + x_2 \leq 2,\quad 10x_1 + 7x_2 \leq 35, \qquad (7.7)$

$$-x_1 - x_2 \le -1, \quad x_1 \ge 0, \quad x_2 \ge 0.$$

Як відомо з аналітичної геометрії, рівняння виду $ax_1 + bx_2 = c$ визначає пряму на площині x_1Ox_2 Рис. 7.1 (якщо a і b не дорівнюють нулю одночасно). При цьому вся площина поділяється на дві півплощини. Нерівність $ax_1 + bx_2 \le c$ описує півплощину, яка лежить з однієї сторони прямої включаючи саму пряму. Ці міркування до помагають зобразити на координатній площині множину точок, які задовольняють умовам функцій обмеження нашого прикладу. Так нерівність $3x_1+5x_2 \le 15$ визначає ту частину півплощини, якій належить початок координат, тому що точка (0,0) їй задовольняє. На рис. 7.1 заштрихована множина точок , що задовольняє усім нерівностям функцій обмеження нашого прикладу, – припустима множина.

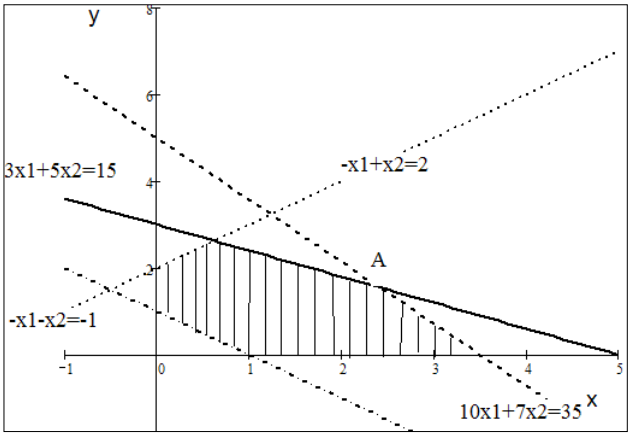


Рисунок 7.1 Графічне розв'язання задачі ЛП . спуску.

Звернемось до цільової функції $x_1 + x_2$. Розглянемо пряму $x_1 + x_2 = 5$ (її графік зображено на рис. 7.1 штриховою лінією). Видно, що її графік не перетинає припустиму множину. Це означає те, що серед припустимих точок (x_1,x_2) не знайдеться таких, для яких цільова функція набула б значення 5.

Рівняння $x_1 + x_2 = d$ за різних значень d описує сімейство прямих. Тоді поставлену нашу задачу лінійного програмування можна переформулювати так: знайти таке максимальне значення *d, за* якого пряма $x_1 + x_2 = d$ перетинає припустиму множину (заштриховану на рис. 7.1). Перша точка на її шляху буде точка A з координатами (70/29, 45/29), яка утворена перетинанням прямих $10x_1 + 7x_2 \le 35$ та $3x_1+5x_2 \le 15$, *тобто максимальне значення цільової функції дорівнює* $F(2{,}41;\ 1{,}55) = 3{,}96$.

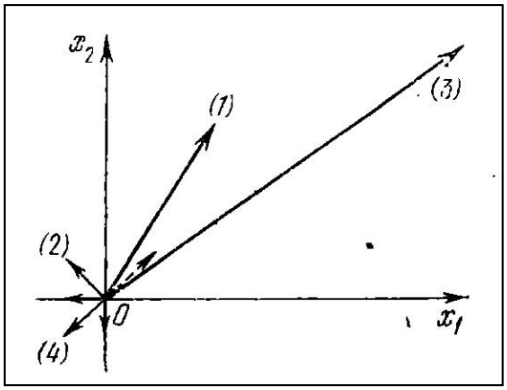


Рисунок. 7.2 – Нормалі до прямих

На цьому прикладі можна побачити усі головні особливості задач лінійного програмування: де припустима множина точок є опуклий багатогранник, який було отримано перетинанням півплощин, а найбільше значення цільової функції досягається у його вершині. Точка A не випадково є розв'язанням нашої задачі. Пригадаємо, що вектор нормалі до прямої $ax_1 + bx_2 = c$ (він же градієнт цієї функції) є вектор з координатами (a,b), тобто у нашому випадку (1,1). Зобразимо вектори нормалей усіх прямих функцій обмеження нашої задачі на координатній площині та штриховою лінією позначимо нормаль цільової функції (рис. 7.2). Вектор нормалі цільової функції лежить по між нормалями до прямих $10x_1 + 7x_2 \le 35$ та $3x_1+5x_2 \le 15$.

ТЕМА 8. СИМПЛЕКС-МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛП

8.1 Симплекс-метод розв'язання задачі лінійного програмування

Симплекс-метод дозволяє знайти розв'язання задачі лінійного програмування за набагато меншу кількість дій ніж простий перебір усіх вершин багатогранника припустимих рішень.

Ідея методу полягає у наступному: знайдемо якусь вершину багатогранника та всі ребра, що виходять з цієї вершини. Підемо вздовж того з ребер, по якому функція зменшується. Прийдемо до наступної вершини, знайдемо усі ребра, які виходять з неї, і повторимо процес. Коли ми прийдемо до такої вершини, що уздовж усіх ребер, які з неї виходять, функція зростає, то мінімуму досягнуто. Оскільки $F(x)$ – лінійна функція, а багатогранник умов опуклий, то цей процес завжди сходиться до вирішення завдання, причому за кінцеве число кроків. За канонічної форми запису багатогранника припустимих рішень з кожної його вершини виходить $N - M$ ребер. Вибираючи одне ребро, ми викидаємо з розгляду вершини, що лежать на інших траєкторіях, отже, за k кроків ми розглядаємо $(N - M)^k$ – у частину вершин, і мінімум досягається приблизно за N кроків.

Виведемо формулу кроку. Першу вершину знаходимо за формулою (7.6). Щоб знайти ребро треба одну з позабазисних змінних x_l зробити позитивною, тоді координати точок ребра можна виразити через неї з (7.6) за допомогою оберненої матриці

$$\tilde{x}_i = x_i - x_l \sum_{j=1}^{M} \alpha_{ij} a_{jl}, \qquad 1 \le i < M, \quad \tilde{x}_l = x_l \tag{8.1}$$

інші позабазисні координати залишаються рівними нулю. Будемо збільшувати x_l до тих пір, доки одна з базисних координат не обернеться у нуль. Це буде при

$$\tilde{x}_l = \min_{1 \le i \le M} \left(\frac{x_i}{S_{il}}\right), \qquad S_{il} = \sum_{j=1}^{M} \alpha_{ij} a_{jl} > 0 \tag{8.2}$$

Мінімум шукаємо лише серед тих індексів i, для яких $S_{il} > 0$, бо тільки ці координати вздовж даного ребра $\tilde{x}_i$ зменшуються і отже, можуть обернутись у нуль. Якщо підставити знайдене у формулу (8.1), отримаємо координати нової вершини і обчислимо в ній значення функції $F(\boldsymbol{x}) = F_l$. По черзі змінюючи кожну позабазисну змінну, знайдемо всі N-M ребер, що виходять з даної вершини і приводять до суміжних вершин. Порівняємо усі значення функції в суміжних вершинах F_l і виберемо з них найменше. Якщо воно менше, ніж значення функції у вихідній вершині F_0, то перемістимося у найнижчу з нових вершин і повторимо процес. Якщо ж $F_l \ge F_0$, то мінімум вже досягнуто у вихідній вершині.

Для виконання усіх кроків та отримання мінімуму знадобиться до $10N^2M^2$ арифметичних дій, що вже прийнятно для сучасних комп'ютерів. Симплекс-метод є прикладом високоспеціалізованого методу. Він придатний лише для знаходження мінімуму лінійної функції в багатовимірному опуклому

багатограннику певного виду − симплексі. Проте дозволяє розв'язати задачу з дуже багатим числом змінних.

8.2 Приклад розв'язування задачі ЛП симплекс-методом

Розглянемо рішення задачі ЛП симплекс-методом на прикладі такої задачі (7.7):

$$\max(x_1 + x_2) \text{ − цільова функція,}$$

функції обмеження $3x_1+5x_2 \le 15,\quad -x_1+x_2 \le 2,\quad 10x_1 + 7x_2 \le 35,$

$$-x_1 - x_2 \le -1,\quad x_1 \ge 0,\quad x_2 \ge 0.$$

Запишемо нашу задачу ЛП у канонічній формі, тобто перетворимо усі нерівності в рівняння за допомогою нових змінних:

$$\begin{aligned} &\max(x_1 + x_2) \\ &3x_1+5x_2+x_3 = 15 \\ &-x_1+ x_2 +x_4 = 2 \\ &10x_1+7x_2 +x_5 = 35 \\ &- x_1 - x_2 +x_6 = -1 \end{aligned} \qquad (8.3)$$

де $x_i \ge 0,\ i = 1,\ldots,6$.

Відсутність нових змінних у цільовій функції означає, що вони входять туди з коефіцієнтами, які дорівнюють 0.

Алгоритм прямого симплекс-методу

1. Вибираємо початкове базисне розв'язання та заповнюємо початкову симплекс-таблицю;
2. Перевіряємо поточне розв'язання на оптимальність, якщо розв'язання оптимальне, то завершуємо розв'язання задачі;
3. Якщо розв'язання не оптимальне, вибираємо напрямом поліпшення розв'язання та визначаємо параметр Θ;
4. Переходимо до нового «поліпшеного» поточного розв'язання.

Повертаємось до пункту 2.

Вибір початкового базисного розв'язання та заповнення початкової симплекс-таблиці

Вибираємо в якості базисних змінних додаткові змінні x_3,x_4,x_5,x_6 , тоді $x_1=x_2=0$. Початкове базисне розв'язання має вид $X^0(0,0,15,2,35,-1)$, цільова функція $F(X^0) = 0$.

Запишемо ці початкові дані у симплекс-таблицю, де позначимо $n = 6$ – число змінних, $m = 4$ − число обмежень.

Симплекс-таблиця має вигляд:

$N_Б$	$C_Б$	$X_Б$	C_i					

					A_{ij}			
		F			Δ_i			

Де $N_Б$ – номера базисних змінних, $C_Б$ – коефіцієнти при базисних змінних, $X_Б$ – значення базисних змінних, C_j – коефіцієнти цільової функції (j = 1,...,6), A_{ij} – коефіцієнти при змінних у рівняннях обмежень (i = 1,…, 4; j = 1,…,6), F – значення цільової функції, Δ_j – рядок оцінок.

Початкова таблиця буде мати вид :

$N_Б$	$C_Б$	$X_Б$	1	2	3	4	5	6
			1	1	0	0	0	0
3	0	15	3	5	1	0	0	0
4	0	2	-1	1	0	1	0	0
5	0	35	10	7	0	0	1	0
6	0	-1	-1	-1	0	0	0	1
		0	-1	-1	0	0	0	0

Оцінка розв’язання для кожного стовпця розраховується за такою формулою:

$$\Delta_j = C_{Бi}A_{ij} - C_j. \qquad (8.4)$$

Так для оцінки Δ_1 треба скалярно помножити стовпець $C_{Бi}$ на стовпець A_{i1} та відрахувати C_1.

Перевірка поточного розв’язання на оптимальність

Критерієм оптимальності або умовою того, що у цій вершині цільова функція має максимальне значення, є виконання умови $\Delta_i \geq 0$, i =1…6. Якщо розв’язується задача ЛП на min умова оптимальності є $\Delta_j \leq 0$, j = 1…6.

При роботі з симплекс-таблицею перевірка розв’язання на оптимальність зводиться до перегляду рядка оцінок Δ_j. *Якщо всі значення рядка оцінок більше або дорівнюють нулю, то поточне розв’язання є оптимальним. Якщо серед значень рядка оцінок є хоча б одне відмінне, то поточне розв’язання не є оптимальним і його треба поліпшувати.*

Перевіримо значення оцінок оптимальності поточного розв’язання нашої таблиці, Δ_1 та Δ_2 мають відмінні значення, отже, поточне розв’язання не є оптимальним та потребує поліпшення.

Вибір напряму поліпшення рішення та визначення параметра Θ

Кількість можливих напрямків поліпшення розв’язання задачі ЛП визначається кількістю відмінних оцінок оптимальності. У нашому прикладі є дві такі оцінки. Кожній оцінці відповідає свій вектор напрямку зростання, для Δ_1 це вектор з координатами A_{i1} , для $\Delta_2 – A_{i2}$. Для поліпшення розв’язання нашої задачі вибираємо один з напрямків, наприклад, для оцінки $\Delta_1 = -1$. Після вибору напрямку зростання перевіряємо критерій необмеженості цільової функції на

області припустимих значень. Він полягає в наступному, якщо серед елементів A_{i1} немає позитивних, то цільова функція необмежена на області припустимих значень і задача ЛП не має розв'язку. Якщо серед елементів A_{i1} є позитивні, розв'язання задачі ЛП можливо поліпшити. У нашому прикладі серед елементів A_{i1} є позитивні.

Тепер нам залишається визначити параметр Θ, який розтягує (звужує) вектор напрямку зростання. Значення цього параметра вибирається за таким правилом:

$$\theta = \min\left(\frac{X_{Бi}}{A_{i1}} \mid A_{i1} > 0 \right). \qquad (8.5)$$

У нашому випадку $\theta = \min\left(\frac{15}{3}, \frac{35}{10}\right) = \min(5,3.5)$. Власне для переходу до нового розв'язання нашої задачі нам потрібен не сам параметр Θ, а номер базисного елемента з мінімальним значенням Θ, тобто $N_Б$.

Параметр Θ може бути використаний для більш точного вибору напрямку зростання цільової функції. Відомо, що скачок цільової функції при переході к «ліпшому» розв'язання дорівнює $\Delta F = -\Delta_j \Theta$. Для того, щоб при поліпшені розв'язання отримати максимальний скачок цільової функції треба розрахувати параметр Θ для всіх від'ємних оцінок Δ_j , визначити ΔF_j і вибрати ту оцінку Δ_j, для якої буде максимальне ΔF_j.

Розглянемо, яку оцінку Δ_j треба вибрати у нашому випадку:

Оцінка $\Delta_1 = -1$, $\Theta_1 = (5,3.5) = 3.5$, $\Delta F_1 = 3.5$;

Оцінка $\Delta_2 = -1$, $\Theta_1 = \min(15/5,2/1,35/7) = 2$, $\Delta F_1 = 2$.

Таким чином, зроблений нами вибір Δ_1, – правильний.

Правило вибору найліпшого напрямку можна сформулювати наступним чином: *вибираємо той напрямок, якому відповідає максимальна абсолютна величина добутку оцінки* Δ_j *на параметр* Θ.

Перехід до нового «поліпшеного» поточного розв'язання

Перехід до нового розв'язання пов'язаний із заміною одних базисних змінних іншими. У симплекс-методі така задача виглядає таким чином:

1) Новою базисною змінною становиться змінна x_1, яка відповідає оцінці Δ_1.
2) З базисних змінних вилучається змінна, яка відповідає рядку, для якого був визначений параметр Θ, тобто x_5.

Нову симплекс-таблицю починаємо заповнювати с того, що замінюємо $N_Б$ та $C_Б$ на нові значення, замість 5 ставимо 1 для стовпця $N_Б$, замість 0 ставимо 1 у стовпця $C_Б$. Елементи C_i залишаються незмінними, як умова задачі. Елементи $X_Б$, A_{ij}, F, Δ_j треба перерахувати методом повного вилучення Гаусса-Жордано з ведучим елементом, який знаходиться на перетині нової базисної змінної (перший стовпець) та рядку з $N_Б = 5$.

$N_Б$	$C_Б$	$X_Б$	1	2	3	4	5	6
			1	1	0	0	0	0

3	0	4.5	0	2.9	1	0	-0.3	0
4	0	5.5	0	1.7	0	1	0.1	0
1	1	3.5	1	0.7	0	0	0.1	0
6	0	2.5	0	-0.3	0	0	0.1	1
		3.5	0	-0.3	0	0	0.1	0

Перерахування методом Гаусса-Жордано означає наступне: ведучий рядок ($N_Б$=5) поділяється на ведучий елемент, інші елементи ведучого стовпця (A_{i1}) заповнюємо нулями. Інші елементи A_{ij} перераховуємо за формулою

$$A_{ij}^{*} = A_{ij} - \frac{A_{3j} A_{i1}}{A_{31}}, j = 1\ldots6,\ i = 1\ldots4. \qquad (8.6)$$

Для розрахунку нових $X_Б$ також використовуємо формулу (8.6), тільки додамо $j = 0$, тобто $A_{i0}^{*} = X_{Бi}$. За допомогою формули (8.4) розрахуємо нові значення оцінок Δ_i.

Ми отримали нове розв'язання. У новому рішенні базисними є змінні x_1, x_3, x_4, x_6 (стовпець $N_Б$), їх значення дорівнюють 4.5; 5.5; 3.5; 2.5 (стовпець $X_Б$). Значення цільової функції дорівнює F = 3.5. Переходимо до наступній ітерації симплекс-методом, тобто повертаємось до кроку 2.

1) Аналіз рядка оцінок показує, що поточне розв'язання не є оптимальним (оцінка $\Delta_2 = -0.3<0$).
2) Тому що відмінна оцінка одна, її вибираємо для визначення напрямку зростання функції. Знаходимо Θ: Θ = min(4.5/2.9; 5.5/1.7; 3.5/0.7) = = 4.5/2.9. Змінна x_2 становиться базисною замість x_3.
3) Переходимо до нового розв'язання, для цього перерахуємо симплекс-таблицю

$N_Б$	$C_Б$	$X_Б$	1	2	3	4	5	6
			1	1	0	0	0	0
2	1	1.55	0	1	0.34	0	-0.10	0
4	0	2.86	0	0	-0.59	1	0.28	0
1	1	2.41	1	0	0.24	0	0.17	0
6	0	2.97	0	0	0.10	0	0.07	1
		3.96	0	0	0.58	0	0.07	0

Елементи рядка оцінок усі додатні, отже ми отримали оптимальне розв'язання. Максимальне значення цільової функції дорівнює F(2.41;1.55) = = 3.96. Це розв'язання для поставленої задачі ЛП ми отримали і графічним методом.

ТЕМА 9. ТРАНСПОРТНА ЗАДАЧА. МЕТОДИ РОЗВ'ЯЗАННЯ ТРАНСПОРТНОЇ ЗАДАЧІ

9.1 Постановка транспортної задачі

Транспортна задача − це одна з типових задач оптимізації, де цільова функція та функції обмеження лінійні, тобто є задачею лінійного програмування. Транспортну задачу формулюють так. Маємо ***m*** **пунктів** виробництва A_i ..., A_m, які мають запаси однорідних продуктів у кількості a_1, ..., a_m одиниць. Маємо ***n*** **пунктів** споживання B_1, ..., B_n, потреби яких у цих продуктах складають b_1, ..., b_n одиниць. Відомі також **транспортні витрати** C_{ij}, перевезення одиниці продукту з пункту A_i в пункт B_j, $i = 1, ..., m; j = 1, ..., n$. Зазвичай повинно виконуватись рівняння

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

тобто загальний обсяг виробництва дорівнює загальному обсягу споживання. Необхідно скласти такий план перевезень (звідки, куди та скільки одиниць продукту перевезти), щоби забезпечити усі пункти споживання B_1, ..., B_m за рахунок реалізації усієї продукції, яку вироблено в пунктах A_i ..., A_m, за мінімальною загальною вартістю усіх перевезень.

Запишемо математичну постановку задачі. Нехай x_{ij} - кількість одиниць продукту, який перевозять з пункту A_i в пункт B_j. Сумарні витрати на перевезення продуктів з усіх пунктів виробництва до всіх пунктів споживання запишемо формулою:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}x_{ij}$$

4) Сумарна кількість продукту, який направлено від кожного пункту виробництва до всіх пунктів призначення, має дорівнювати запасу продукту в цьому пункті, тобто

$$\sum_{j=1}^{n} x_{ij} = a_i \quad , i = 1, ..., m.$$

Сумарна кількість вантажу, який перевозять до кожного пункту споживача з усіх пунктів відправлення, має дорівнювати потребам. Це умова повного задоволення потреб

$$\sum_{i=1}^{m} x_{ij} = b_j \quad , \ j = 1, \ldots, n.$$

Обсяги перевезень − невід'ємні числа, тому що перевезення з пунктів споживання до пунктів виробництва не можливі:

$$x_{ij} \geq 0, i = 1, ..., m; j = 1, ..., n.$$

Таким чином, транспортна задача − це визначення мінімальних сумарних витрат на перевезення вантажів при виконанні умов повного задоволення попиту та рівняння вивезеної кількості продукту до його запасів у пунктах відправлення.

Для визначення оптимального плану транспортної задачі можна застосовувати симплекс-метод. Проте завдяки специфіки обмежень транспортної задачі (кожна невідома входить лише у два рівняння системи обмежень і коефіцієнти при невідомих дорівнюють одиниці) для визначення оптимального плану транспортної задачі розроблені спеціальні методи.

Як і при розв'язанні задачі лінійного програмування симплексним методом, визначення оптимального плану транспортної задачі починають із знаходження будь-якого її опорного плану. *Опорний план* – це будь-яке допустиме розв'язання транспортної задачі. Для визначення опорного плану вихідні дані транспортної задачі записують у таблицю умови.

9.2 Чисельні методи розв'язання транспортної задачі

Розглянемо чисельні методи розв'язання транспортної задачі на наступному прикладі: на складах виробника $A1$, $A2$, $A3$ зберігають $a1 = 100$, $a2 = 200$, $a3 = 120$ одиниць одного й того самого вантажу. Треба доставити його трьом споживачам $B1$, $B2$, $B3$, замовлення яких складають $b1 = 200$, $b2 = 110$, $b3 = 80$ одиниць вантажу відповідно. Вартість перевезення $C_{i,j}$ одиниці вантажу з i-го складу j-му споживачеві наведено у таблиці:

	$b1 = 200$	$b2 = 110$	$b3 = 80$
$a1 = 100$	4	2	6
$a2 = 200$	7	5	3
$a3 = 120$	1	7	6

Необхідно знайти мінімальну вартість перевезень. Позначимо кількість вантажу, який перевозиться з i-го складу до j-го споживача як x_{ij}. Перевіримо баланс задачі, тобто кількість вантажу, яка є на складах A_i повинна дорівнювати кількості вантажу, яку потребують споживачі B_j. У нашому випадку $\sum_{i=1}^{3} a_i = 100 + 200 + 120 = 420$ та $\sum_{j=1}^{3} b_j = 200 + 110 + 80 = 390$ ця умова не виконується.

Треба збалансувати нашу задачу. Для цього, понад наявних n пунктів призначення $b1$, $b2$, $b3$ уведемо ще один, фіктивний, пункт призначення $b4$, якому припишемо фіктивну заявку, значення якої дорівнює надлишку запасів над заявками. Вартість перевезень з усіх пунктів відправлення до фіктивного пункту призначення $b4$ вважатимемо нульовою. Введенням фіктивного пункту призначення $b4$ з його заявкою $b4$ ми зрівняли баланс транспортної задачі й тепер її можна розв'язувати як звичайну транспортну задачу із правильним балансом. Тепер вихідні дані запишемо в таблицю умови:

	$b1$	$b2$	$b3$	$b4$	Вироб.
$a1$	4 x_{11}	2 x_{12}	6 x_{13}	0 x_{14}	100
$a2$	7 x_{21}	5 x_{22}	3 x_{23}	0 x_{24}	200
$a3$	1 x_{31}	7 x_{32}	6 x_{33}	0 x_{34}	120
Спож.	200	110	80	30	420

Цільова функція має вид:

$F(X)= 4\,x_{11}+2\,x_{12}+6\,x_{13}+7\,x_{21}+5\,x_{22}+3\,x_{23}+x_{31}+7\,x_{32}+6\,x_{33}$

Функції обмеження:

$x_{11}+x_{12}+x_{13}+x_{14}=100$
$x_{21}+x_{22}+x_{23}+x_{24}=200$
$x_{31}+x_{32}+x_{33}+x_{34}=120$
$x_{11}+x_{21}+x_{31}=200$
$x_{12}+x_{22}+x_{32}=110$
$x_{13}+x_{23}+x_{33}=80$
$x_{14}+x_{24}+x_{34}=30 \qquad x_{ij}\geq 0,\ i=1,\ldots,3;\ \ j=1,\ldots,4.$

З таблиці умови випливає, що будь-яке обмеження транспортної задачі є лінійною комбінацією інших обмежень. Отже система обмежень транспортної задачі лінійно залежна та містить тільки $m+n-1$ незалежне рівняння.

Сутність викладених нижче чисельних методів розв'язання транспортної задачі полягає в тому, що опорний план знаходять послідовно за $n+m-1$ кроків, на кожному з яких в таблиці умови завдання заповнюють одну клітинку, яку називають зайнятою. Заповнення однієї з клітинки забезпечує повністю або задоволення потреби у вантажі одного з пунктів призначення (того, в стовпці якого знаходиться заповнена клітинка), або вивіз вантажу з одного з пунктів відправлення (того, в рядку якого знаходиться і заповнюється клітинка). У першому випадку тимчасово вилучають з розгляду стовпець, що містить заповнену на даному кроці клітинку, і розглядають задачу, таблиця умови якої містить на один стовпець менше, ніж було перед цим кроком, але ту саму кількість рядків і відповідно змінення запасів вантажу в одному з пунктів відправлення (у тому, за рахунок запасу якого була задоволена потреба у вантажі в пункті призначення на даному кроці). У другому випадку тимчасово вилучають з розгляду рядок, що містить заповнену на даному кроці клітинку, і розглядають задачу, таблиця умови якої містить на один рядок менше, ніж було перед цим кроком, але ту саму кількість стовпців і відповідно змінення потреби у вантажі в одному з пунктів призначення, в стовпці якого заповнюється клітинка.

Після того як виконані $m+n-2$ описані вище кроки, отримують завдання за одним пунктом відправлення і одним пунктом призначення. При цьому залишається вільною лише одна клітинка, а запаси , які залишились у пункті відправлення дорівнюватимуть потребам, які залишилися у пункті призначення. Коли заповнюють цю клітинку, тим самим роблять $(m+n-1)$-й крок, і отримують вихідний план.

Окремо слід розглянути випадок, коли на деякому кроці (але не на останньому) може виявитися, що потреби чергового пункту призначення дорівнюють запасам чергового пункту відправлення. В цьому випадку також тимчасово вилучають з розгляду або стовпець, або рядок (щось одне). Таким чином, або запаси відповідного пункту відправлення, або потреби даного пункту призначення вважаються рівними нулю. Цей нуль записують у чергову клітину, яку треба заповнити. Зазначені вище умови гарантують отримання

$m + n - 1$ зайнятих клітинок, в яких стоять компоненти опорного плану, що є вихідною умовою для перевірки останнього на оптимальність і знаходження оптимального плану.

Розглянемо цей алгоритм на прикладі обліку опорного плану методом північно-західного кута. На кожному кроці методу північно-західного кута з усіх не викреслених клітинок вибираємо саму ліву та верхню клітинку. Знаходимо значення x_{11} з відношення $x_{11} = \min\{a_1, b_1\}$.

Можливі три варіанти:

- Якщо $a_1 < b_1$, то $x_{11} = a_1$, *рядок* $i = 1$ вилучається з подальшого розгляду, а потрібність першого споживача b_1 $(j = 1)$ зменшується на величину a_1;
- Якщо $a_1 > b_1$, то $x_{11} = b_1$, *стовпець* $j = 1$ вилучається з подальшого розгляду, а наявність вантажу на першому складі a_1 $(i = 1)$ зменшується на величину b_1;
- Якщо $a_1 = b_1$, то $x_{11} = a_1 = b_1$, *рядок* $i = 1$ *та стовпець* $j = 1$ вилучаються з подальшого розгляду. Цей варіант дає виродження вихідного плану.

Далі аналогічні операції виконуються з іншою частиною таблиці, починаючи з її північно-західного кута. Після заповнення клітинки, яка знаходиться на перетині, процес закінчується.

Після завершення описаного процесу необхідно провести перевірку отриманого плану на виродженість. Якщо кількість заповнених клітинок дорівнює $m + n - 1$, то план є невиродженим, в іншому разі – виродженим.

Якщо план вироджений, тобто кількість заповнених клітинок виявилась меншою $m+n-1$, то незаповнені клітинки з мінімальною вартістю перевезення заповнюються нулями, щоб спільна кількість заповнених клітинок дорівнювала $m+n-1$. Проте при розстановці нулів необхідно пам’ятати, що у таблиці не повинно бути жодного прямокутника, всі вершини якого є заповнені клітинки. Наприклад, змінні $x_{11}, x_{12}, x_{21}, x_{22}$ або $x_{11}, x_{14}, x_{21}, x_{24}$ не можуть бути одночасно базисні.

1. $x_{11} = \min\{100, 200\} = 100$, рядок 1 вилучається з подальшого розгляду, кількість вантажу для першого споживача зменшується на 100 і дорівнює 100.
2. $x_{21} = \min\{200, 100\} = 100$, стовпець 1 вилучається з подальшого розгляду, кількість вантажу на другому складі зменшується на 100 і дорівнює 100.
3. $x_{22} = \min\{100, 110\} = 100$, рядок 2 вилучається з подальшого розгляду, кількість вантажу для другого споживача зменшується на 100 і дорівнює 10.
4. $x_{32} = \min\{120, 10\} = 10$, стовпець 2 вилучається з подальшого розгляду, кількість вантажу на третьому складі зменшується на 10 і дорівнює 110.
5. $x_{33} = \min\{110, 80\} = 80$, стовпець 3 вилучається з подальшого розгляду, кількість вантажу на третьому складі зменшується на 80 і дорівнює 30.

Оскільки у таблиці залишився один стовпець 4 у третьому рядку – це останній крок процесу, $x_{34} = 30$.

	*b*1	*b*2	*b*3	*b*4	Вироб.

*a*1	4 *100*	2	6	0	100
*a*2	7 100	5 100	3	0	200/100
*a*3	1	7 10	6 80	0 30	120/110/30
Спож.	200/100	110/10	80	30	420

Таким чином, ми отримали опорний план:

$$X=\begin{pmatrix} 100 & 0 & 0 & 0 \\ 100 & 100 & 0 & 0 \\ 0 & 10 & 80 & 30 \end{pmatrix}$$

Далі цей опорний план треба оптимізувати.

Поліпшення опорного плану методом потенціалів

Загальний принцип визначення оптимального плану транспортної задачі методом потенціалів аналогічний принципу розв'язання задачі лінійного програмування симплекс-методом, а саме: спочатку знаходять опорний план транспортної задачі, а потім його послідовно покращують до отримання оптимального плану.

Теорема. Якщо для деякого опорного плану $X^* = (x_{ij}^*)$ $(i = 1,...,m;\ j = = 1, ...,n)$ транспортної задачі існують такі числа $\alpha_1, \alpha_2, \ldots, \alpha_m, \beta_1, \beta_2, \ldots, \beta_n$, що

$$\beta_j+\alpha_i=c_{ij} \text{ при } x_{ij} > 0$$

$$\beta_j-\alpha_i <= c_{ij} \text{ при } x_{ij} = 0$$

для всіх ($i = 1, m$) і ($j = 1, n$), то $X^* = (x_{ij}^*)$ - оптимальний план транспортної задачі. Числа α_i і β_j ($i = 1, m; j = 1, n$) називаються *потенціалами* відповідно до пунктів призначення та пунктів відправлення.

Сформульована теорема дозволяє побудувати алгоритм знаходження розв'язання транспортної задачі. Він полягає в наступному. Нехай одним з розглянутих вище методів знайдено опорний план транспортної задачі. Для кожного з пунктів відправлення та призначення визначаються потенціали α_i і β_j $(i = 1,...,m;\ j =1,...,n)$. Ці числа знаходять із системи рівнянь $\beta_j+\alpha_i = c_{ij}$де c_{ij} − тарифи, які стоять в заповнених клітинках таблиці умов транспортної задачі.
Через те що число заповнених клітинок дорівнює $m+n-1$, то система з $m+n$ невідомими містить $m+n-1$ рівнянь. Оскільки число невідомих перевищує на одиницю число рівнянь, одне з невідомих можна покласти рівним безпідставному числу, наприклад $\alpha_1 = 0$, і знайти послідовно з рівнянь значення інших невідомих. Після того як усі потенціали знайдені, для кожної з вільних клітинках визначають числа $a_{ij} = \beta_j - \alpha_i - c_{ij}$. *Якщо серед чисел a_{ij} $(i = 1, m; j = 1,n)$ немає позитивних, то знайдений опорний план є оптимальним.* Якщо ж для деякої вільної клітинки $a_{ij}>0$, то вихідний опорний план не є оптимальним і необхідно перейти до нового опорного плану. Для цього розглядають усі вільні клітинки, для яких $a_{ij}>0$, і серед даних чисел вибирають максимальне. Клітинку, якій відповідає це число, слід заповнити.

Заповнюючи обрану клітинку, необхідно змінити обсяги поставок, записаних в інших зайнятих клітинок і пов'язаних із заповнюваної клітинкою

так званим циклом. *Циклом* в таблиці умов транспортної задачі називається ламана лінія, вершини якої розміщені в зайнятих клітинках таблиці, а ланки - уздовж рядків і стовпців, причому в кожній вершині циклу зустрічається рівно дві ланки, одна з яких знаходиться в рядку, а інша в стовпці.

Якщо ламана лінія, що утворює цикл, перетинається, то точки самоперетинання не є вершинами. Приклади деяких циклів показані на рис. 9.1

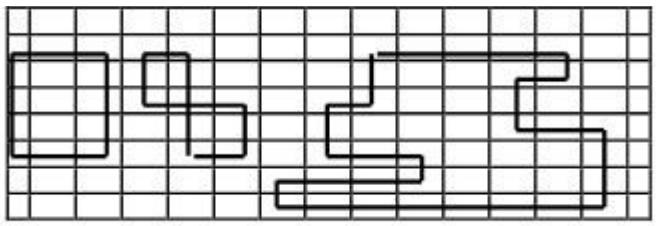


Рисунок 9.1− Приклад циклів у методі потенціалів

При правильній побудові опорного плану для будь-якої вільної клітинки можна побудувати лише один цикл. Після того як для вибраної вільної клітинки він побудований, слід перейти до нового опорного плану. Для цього необхідно перемістити вантажі в межах клітинок, пов'язаних циклом з даною вільної клітинкою. Це переміщення виконують за такими правилами:

1) кожній з клітинок, пов'язаних циклом з даною вільною клітинкою, приписують певний знак, причому вільній клітинці − знак плюс, а всім іншим клітинкам − по черзі знаки мінус і плюс (будемо називати ці клітинки мінусовими і плюсовими);

2) в дану вільну клітинку переносять менше з чисел x_{ij}, що стоять в мінусових клітинках. Одночасно це число додають до відповідних чисел, що стоять в плюсових клітинках, і віднімають з чисел, що стоять у мінусових клітинках. Клітинка, яка раніше була вільною, ставати зайнятою, а мінусова клітинка, в якій стояло мінімальне з чисел x_{ij}, вважається вільною.

У результаті зазначених вище переміщень вантажів у межах клітинок, пов'язаних циклом з даною вільної клітинкою, визначають новий опорний план транспортної задачі. Описаний вище перехід від одного опорного плану транспортної задачі до іншого її опорному плану називається *зсувом за циклом перерахунку*.

Слід зазначити, що при зсуві за циклом перерахунку число зайнятих клітинок залишається незмінним, а саме залишається рівним $m+n-1$. При цьому якщо в мінусових клітинках є два (або більше) однакових числа x_{ij}, то звільняють лише одну з таких клітинок, а інші залишаються зайнятими (з нульовими поставками).

Отриманий новий опорний план транспортної задачі перевіряють на оптимальність. Для цього визначають потенціали пунктів відправлення та призначення і знаходять числа $a_{ij} = \beta_j - \alpha_i - c_{ij}$ для всіх вільних клітинок. Якщо серед цих чисел не виявиться позитивних, то це свідчить про отримання оптимального плану. Якщо ж позитивні числа є, то слід перейти до нового

опорного плану. В результаті ітераційного процесу після кінцевого числа кроків отримують оптимальний план задачі.

Розглянемо метод потенціалів на прикладі нашої транспортної задачі.

	$b1$ /β_1=4	$b2$/β_2 =2	$b3$/β_3 =1	$b4$ /β_4=-5	*Вироб.*
$a1$ /α_1=0	4 *100*	2	6	0	100
$a2$ /α_2=3	7 100	5 100	3	0	200
$a3$ /α_3=5	1	7 10	6 80	0 30	120
Спож.	200	110	80	30	420

Розрахуємо потенціали та числа $a_{ij} = \beta_j - \alpha_i - c_{ij}$:

$\alpha_1+ \beta_1= 4$; $\alpha_1+ \beta_2= 2$; $\alpha_2+ \beta_3= 3$; $\alpha_2+ \beta_2= 5$; $\alpha_3+ \beta_2= 7$; $\alpha_3+ \beta_3= 6$

$\beta_1= 4$; $\beta_2= 2$; $\alpha_2= 3$; $\alpha_3= 5$; $\beta_3= 1$; $\beta_4= -5$

$a_{12}= 0$; $a_{13}= -5$; $a_{14}= -5$; $a_{23}= -5$; $a_{24}= -8$; $a_{31}= -2$

Критерій оптимальності виконано, тому що усі $a_{ij} \le 0$. Наш опорний план є оптимальним.

Розв’язанням нашої транспортної задачі є матриця

$$X=\begin{pmatrix} 100 & 0 & 0 & 0 \\ 100 & 100 & 0 & 0 \\ 0 & 10 & 80 & 30 \end{pmatrix},$$

значення цільової функції дорівнює $F(X) = 2150$.

9.3 Розв’язування транспортної задачі в Excel

Для розв'язання таких завдань в Excel існує спеціальний засіб *Поиск решения*. Але перед тим як використовувати його, потрібно ввести вихідні дані. Розрізняють такі дані оптимізаційної задачі: *параметри управління, цільова функція та обмеження:*

1) ***Для параметрів управління*** потрібно відвести область комірок, де вони будуть записуватись. Потім в цю область слід, ввести довільні значення параметрів (наприклад, усі одиниці). Під час роботи *Поиск решения* підбиратиме значення цих параметрів доти, поки не отримає оптимальний план.

2) ***Цільову функцію будують,*** використовуючи посилання на комірки з початковими значеннями параметрів управління. Комірку, де міститься формула цільової функції, називають цільовою.

3) ***Кожне обмеження*** задачі в математичному записі має такий вигляд: *hRb*; де *H* − деяка функція, *R*− одне з відношень =, > або <; *b* − дійсне число, або посилання на комірку. Формулу функції *h* (ліва частина обмеження) та значення *b* (права частина) потрібно ввести у дві комірки, а запис обмеження відбувається безпосередньо у *Поиск решения.*

Інсталяція засобу Поиск решения

Якщо в меню *Сервис* немає пункту *Поиск решения,* цей засіб потрібно інсталювати.

- **Для інсталяції засобу** *Поиск решения* виконайте команду *Сервис* ▶ *Надстройки.*

Потім у групі *Список надстроек* виберіть пункт *Поиск решения.* Після цього натисніть кнопку *ОК* .

Засіб *Поиск решения*

- Для запуску засобу *Поиск решения* виконайте команду *Сервис* ▶ *Поиск решения.*

- Діалогове вікно *Поиск решения* містить три групи полів, які потрібно заповнити. Це опції для цільової комірки, адреси комірок із змінними параметрами керування та поле обмежень (рис. 9.2).

- У групі опцій цільової комірки зазначте адресу комірки (у полі *Установить целевую ячейку)* і тип оптимізаційної задачі. У підгрупі *равной* (тип задачі) є такі пункти:

–*максимальному значенню* (задача максимізації цільової функції);
–*минимальному значенню* (задача мінімізації цільової функції);
–*значенню* (задача рівності цільової функції конкретному числу).

За замовчуванням цільовою вважається комірка, де розміщується курсор.

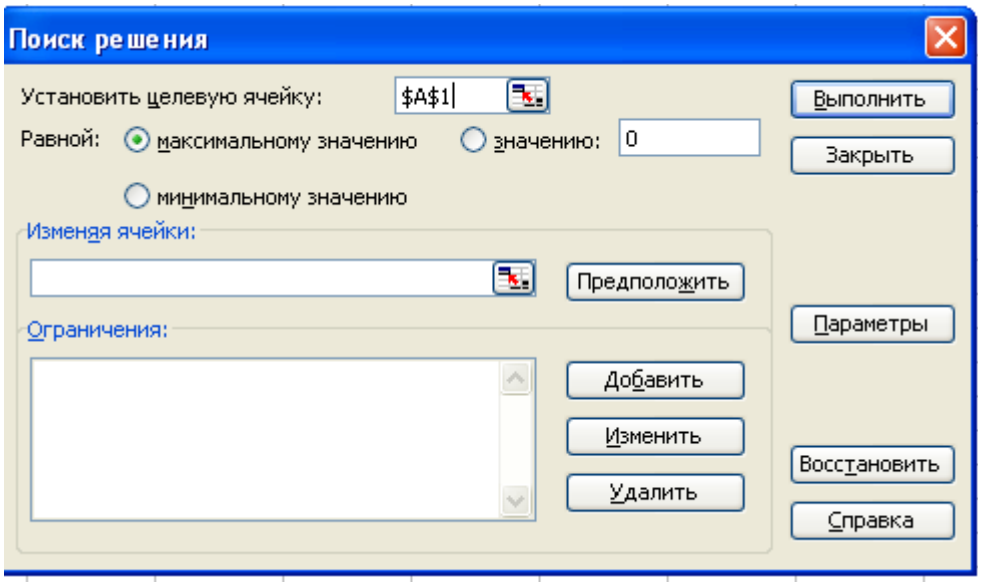


Рисунок 9.2 – Вікно діалогу Поиск решения

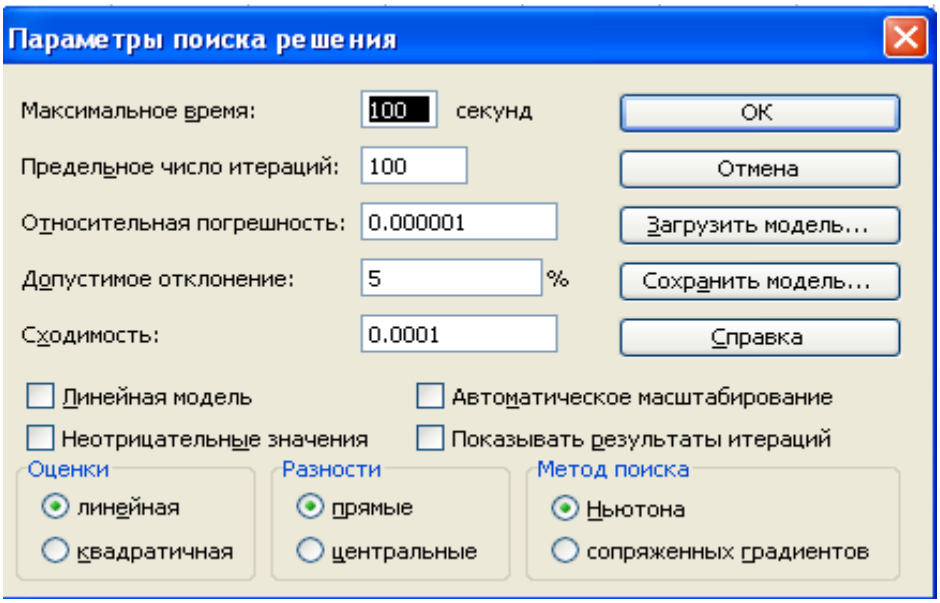


Рисунок 9.3 – Вікно параметрів пошуку рішення

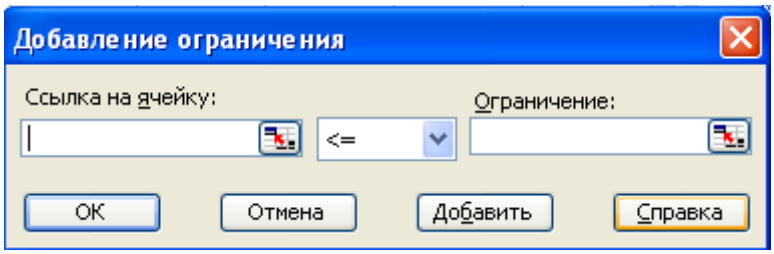


Рисунок 9.4 - діалогове вікно *Добавление ограничения*

Адреси змінних параметрів управління задачі задайте в полі *Изменяя ячейки.* Обмеження задачі задаються у групі *Ограничения.* Їх вводять натисканням кнопки *Добавить.* Ця кнопка викликає діалогове вікно *Добавление ограничения (див. рис. 9.4),* де потрібно заповнити три поля. У лівому полі *Ссылка на ячейку* зазначають адресу лівої частини обмеження, у правому полі *Ограничение* – адресу правої частини обмеження або число, з яким порівнюється ліва частина, і в центральному полі вибирають тип обмеження: <=, =, >=, *"цел"* або *"двоич".* Останні два типи вказують, що ліва частина набуває лише цілих значень або відповідно значення 0 і 1. Обмеження додають до списку обмежень, використовуючи кнопку *Добавить* або *ОК.* При цьому:

- кнопка *Добавить* дає змогу записати наступне обмеження;
- кнопка *ОК* закриває вікно *Добавление ограничения.*

Редагувати обмеження можна за допомогою кнопки *Изменить.* Параметри керування засобом *Поиск решения* задають у діалоговому вікні *Параметры поиска решения,* що викликається натисканням кнопки *Параметры* (рис. 9.3). Розглянемо параметри керування (їх значення наведено в дужках за замовчуванням).

- *Максимальное время* (100 с) – максимальний час, відведений на розв'язування задачі. Якщо за цей час *Поиск решения* не знайде оптимального розв'язку, він повідомить результати останньої ітерації.

- *Предельное количество итераций* (100) − обмеження на час роботи *Поиск решения* у термінах максимальної кількості ітерацій алгоритму.

- *Относительная погрешность* (0,000001) − відносна точність, з якою шукається оптимальне значення цільової комірки.

- *Допустимое отклонение* (5 %) – допустиме відхилення значення цільової комірки від оптимального, якщо в задачі є параметри, область зміни яких обмежена цілими числами.

- Параметр *Сходимость* (0,0001). Якщо відносна зміна у п'яти останніх ітераціях менша від цього параметра, оптимізаційна задача вважається розв'язаною. Цей параметр можна застосувати тільки для нелінійних задач.

- Параметр *линейная модель* використовує методи лінійного програмування.

- Параметр *значения не отрицательны* означає, що всі змінні параметри невід'ємні.

- Параметр *автоматическое масштабирование* використовують тоді, коли значення змінних параметрів та оптимальне значення цільової комірки істотно різняться.

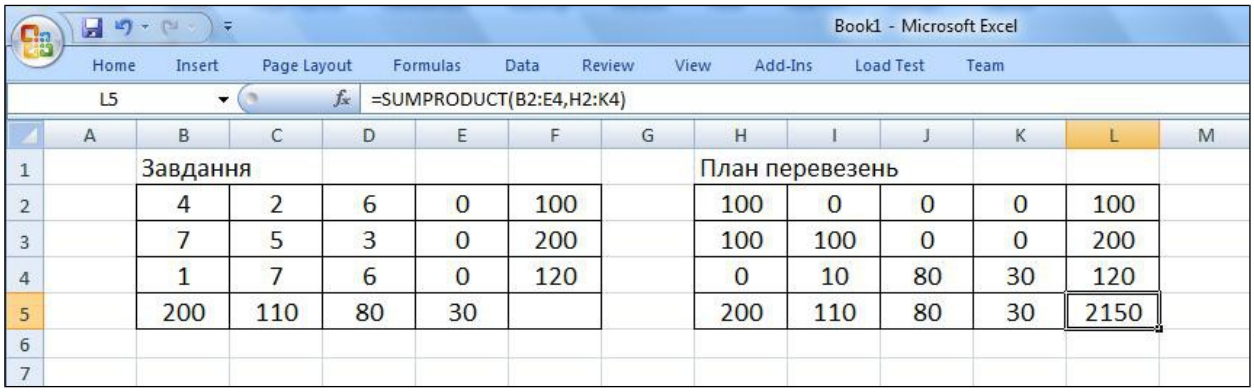
- Параметр *показывать результати итераций* виводить проміжні результати після кожної ітерації.

-У нелінійних задачах у групі *Оценки* доцільніше вибрати опцію *квадратичная.*

- Параметри *разности* і *Метод поиска.* Доцільніше залишити значення параметрів цих груп, що є за замовчуванням.

-Для розв'язання задачі натисніть кнопку *Выполнить.*

Розв’язання транспортної задачі, яка наведена у пункті 9.4, у Excel подано на рис. 9.5.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Завдання						План перевезень					
2		4	2	6	0	100		100	0	0	0	100	
3		7	5	3	0	200		100	100	0	0	200	
4		1	7	6	0	120		0	10	80	30	120	
5		200	110	80	30			200	110	80	30	2150	
6													
7													

Рисунок 9.5 – Приклад розв’язання транспортної задачі в Excel

Аналіз результатів

Після визначення розв'язання оптимізаційної задачі засіб *Поиск решения* відкриває діалогове вікно *Результати поиска решения,* звідки вибирають бажані типи звітів про розв'язання.

- Для простого відображення розв'язання в робочому аркуші виберіть опцію *Сохранить найденное решение.*

- Для відмови від отриманого розв'язання виберіть опцію *Восстановить исходные значения.*

- Для відображення результатів на окремому аркуші виберіть тип звіту *Результати.* При цьому на новому аркуші буде наведено інформацію про оптимальний план та оптимальне значення параметрів, а також про зв'язаність (рівність лівої та правої частин обмеження) чи незв'язаність обмежень.

- Для отримання звіту про стійкість розв'язання щодо малих змін у цільовій функції та обмеженнях виберіть тип звіту *Устойчивость.* Зауважимо, що найважливішим результатом звіту про стійкість є *множники Лагранжа* (тіньові ціни). Множник Лагранжа для кожного обмеження вказує на миттєве покращення значення цільової функції, якщо збільшити (за умови відношення "= <") праву частину обмеження на 1.

- Для аналізу допустимих змін кожного параметра за умови, що значення інших параметрів є фіксованими і такими, як в оптимальному плані, використовують тип звіту *Предельный.*

- За допомогою миші можна вибрати кілька типів звітів одночасно.

Моделі оптимізації

Якщо на одному аркуші потрібно розв'язати кілька різних оптимізаційних задач, з кожною задачею пов'язують модель оптимізації. Модель оптимізації містить інформацію про цільову комірку, тип задачі, її змінні параметри, обмеження та параметри алгоритму. Для даних кожної моделі в аркуші потрібно відвести місце. Автоматично Excel запам'ятовує лише одну (першу) модель у кожному аркуші.

- Для збереження поточної моделі (що міститься в даний момент у вікні *Поиск решения)* потрібно послідовно натиснути кнопки *Параметри, Сохранить модель* і зазначити область, де зберігатиметься модель.

- Для завантаження іншої моделі (що була записана раніше) слід послідовно натиснути кнопки *Параметри, Загрузить модель* і зазначити адресу моделі.

ТЕМА 10. ОСНОВИ ТЕОРІЇ ГРАФІВ. ПОДАННЯ ГРАФІВ У КОМП'ЮТЕРІ. АЛГОРИТМИ ПЕРЕГЛЯДУ ВЕРШИН ГРАФІВ

10.1 Основні поняття теорії графів

Теорія графів – розділ дискретного аналізу, який набув широкого застосування у багатьох наукових дисциплінах завдяки тому, що поняття та інструментарій цієї теорії виявився дуже зручним для дослідження та інтерпретації різноманітних проблем у значній кількості наук: кібернетиці, технічній і економічній, теорії автоматів, теорії управління, теорії інформації тощо. За допомогою теорії графів розв'язується значна кількість економіко-математичних задач, зокрема задача "про призначення", задачі календарного планування, мережного планування тощо. Взагалі теорія графів має велике значення у телекомунікаціях.

Нехай V – довільна множина, E – деяка сукупність пар вигляду (v_i, v_j), де v_i , v_j є V. Упорядкована пара G=(V,E), яка складається з множини V та сукупності E, називається графом із множиною вершин V і множиною ребер E.

Іншими словами, графом називається пара двох множин: вершин *V* і ребер *E*, що зв'язують вершини. Термін "сукупність" означає можливість наявності однакових пар.

При графічному зображенні графа (рис. 10.1) елементи множини *V* зображають точками на площині,

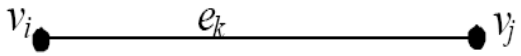


Рисунок 10.1 − Зображення графа

а ребра *(v_i, v_j)* – відрізками (прямолінійними або криволінійними), які з'єднують точки v_i і v_j. Ребра звичано позначаються e_k . Ребро e_k називається ***інцидентним*** по відношенню до вершин v_i і v_j . Граф називається ***скінченим***, якщо множини його вершин і ребер є скінченими. Множину вершин графа *G* позначають *V(G)*, а множину ребер – *E(G)*. Кількість вершин графа – *n(G)*, а кількість ребер графа – *m(G)*. *Кількість вершин n(G) графа називають його* ***порядком***.

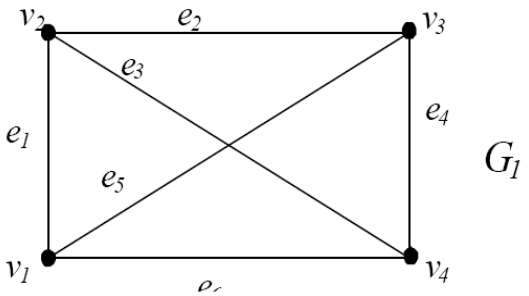


Рисунок 10.2− Приклад скінченого графа

Граф *G1*(рис. 10.2) має множину вершин *V(G1) = {v1, v2, v3, v4}* і множину ребер *E(G1) = {e1, e2, e3, e4, e5, e6}*. *n(G1) = 4*, тобто порядок графа *G1* дорівнює 4, *m*(*G1)*=6. В графі *G1* суміжними є всі вершини v*1, v2, v3, v4*, суміжні трійки ребер *e1, e2, e3*; *e1, e5, e6*; *e6, e3, e4*; *e2, e3, e4*. Розглянемо ребро *e2*. Вершини v*2* і v*3* є кінцями ребра *e2*. Вершини v*2* і v*3* інцидентні ребру *e2*. Ребро *e2* інцидентне вершинам v*2* і v*3*.

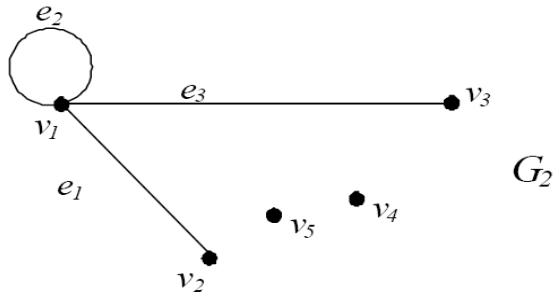


Рисунок 10.3− Граф незв'язаними вершинами

Граф *G2* (рис. 10.3) має множину вершин *V(G2) = {v1, v2,v3, v4, v5}* і множину ребер *E(G2) = {e1, e2, e3)*. *n*(*G2)*=5, *m*(*G1)*=3. Це граф порядку 5. В графі *G2* не всі вершини суміжні між собою. Вершини v*4* і v*5* не суміжні жодній вершині графу. Ребро *e3* інцидентне вершинам v*1* і v*3*, а вершини v*1* і v*3* інцидентні ребру *e3*. *r*(*v*1) = *1*, *r*(*v*3) = *1*, *r*(*v*2) = *1*.

Ребра *e*1, *e*2, *e*3− суміжні. Якщо пари (v_i,v_j) ϵ *E* вважаються упорядкованими, то граф називається ***орієнтованим*** *або* ***орграфом***. Інакше граф називається ***неорієнтованим***. Ребра орієнтованого графа прийнято називати ***дугами*** та зображати направленими відрізками. При зображенні орієнтованих графів напрямки ребер позначаються стрілками (рис. 10.4). Орієнтований граф також може мати кратні ребра, петлі, а також ребра, що з'єднують одні й ті самі вершини ребра, але у зворотних напрямках.

V={*x*1*;x*2*;x*3*;x*4*;x*5}; *E*={(*x*1;*x*2); (*x*2;*x*4); (*x*1;*x3*); (*x*2;*x*3); (*x*3;*x*4); (*x*4;*x*5)}.

Наприклад: *l* = (*x*4;*x*5) то *x4* – початок дуги, *x5* – кінець. Задати граф означає задати множини його вершин і ребер, а також відношення інцидентності. Розглянемо звичайні графи. Нехай – $\{v_1,v_2,...,v_n\}$ вершини графа *G*, $\{e_1,e_2,....,e_n\}$ – його ребра. Відношення інцидентності можна позначити матрицею *E*={ε_{ij}}, яка має *m* рядків і *n* стовпців (рис. 10.5), де

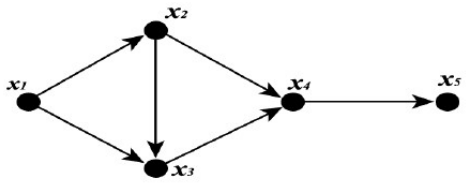
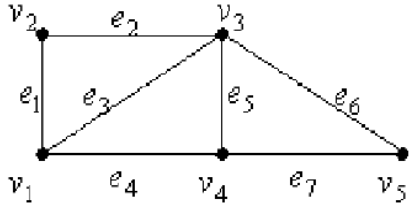


Рисунок 10.4− Приклад орієнтованого графа

$$\varepsilon_{ij} = \begin{cases} 1, якщо \cdot ребро \cdot e_i \cdot інцидентне \cdot вершині \cdot v_j \\ 0, в \cdot протилежному \cdot випадку \end{cases}$$

Стовпці матриць відповідають вершинам графа, а рядки – його ребрам.



Матриця інцидентості

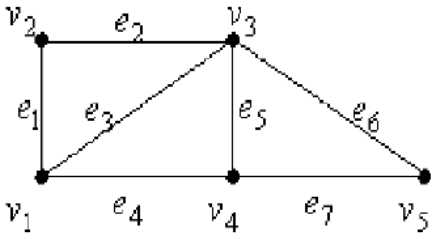
	v_1	v_2	v_3	v_4	v_5
e_1	1	1	0	0	0
e_2	0	1	1	0	0
e_3	1	0	1	0	0
e_4	1	0	0	1	0
e_5	0	0	1	1	0
e_6	0	0	1	0	1
e_7	0	0	0	1	1

Рисунок 10.5 - Матриця інцидентності

Матриця суміжності графа (рис. 10.6) − це квадратна матриця Δ={δ_{ij}}, стовпцям і рядкам якої відповідають вершини графа. Для неорієнтованого

графа δ_{ij} дорівнює кількості ребер, інцидентних *i*-й і *j*-й вершинам, для орієнтованого графа цей елемент матриці суміжності відповідає кількості ребер з початком у *i*-й вершині й кінцем у *j*-й. Таким чином, матриця суміжності неорієнтованого графа симетрична ($\delta_{ij} = \delta_{ji}$), а орієнтованого – необов'язково.

$$\delta_{ij} = \begin{cases} 1, якщо \cdot вершини \cdot з \cdot даними \cdot номерами \cdot суміжні; \\ 0, в \cdot протилежному \cdot випадку. \end{cases}$$



Матриця суміжності

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	1	0
v_2	1	0	1	0	0
v_3	1	1	0	1	1
v_4	1	0	1	0	1
v_5	0	0	1	1	0

Рисунок 10.6 − Матриця суміжності

Матриця суміжності повністю визначає відповідний неорієнтований або орієнтований граф. Число його вершин дорівнює розмірності матриці.

Нехай G – неорієнтований граф. ***Маршрутом*** *M у графі G, що з'єднує вершину* $\boldsymbol{v_1}$ *з вершиною* $\boldsymbol{v_n}$ *, називається така послідовність вершин і ребер, які чергуються* $M=\{ \boldsymbol{v_1, e_1, v_2, e_2, \ldots, v_{n-1}, e_{n-1}, v_n}\}$, *що починається у вершині* $\boldsymbol{v_1}$ *і закінчується у вершині* $\boldsymbol{v_n}$, *і така, що кожні два сусідні ребра* e_{i-1} *і* e_i *мають спільну інциденту вершину.*

Очевидно, що маршрут *M* можна задавати послідовністю його вершин $\{v_1, v_2, \ldots, v_n\}$ (у звичайному графі), а також послідовністю ребер $\{e_1, e_2, \ldots, e_n\}$ (для будь-якого графа). Одне і те саме ребро може зустрічатися в маршруті кілька разів. Вершина $\boldsymbol{v_1}$ називається початком маршруту, $\boldsymbol{v_n}$ – кінець маршруту. Число ребер маршруту називається його ***довжиною***. Якщо $\boldsymbol{v_1 = v_n}$, то маршрут називають ***замкненим***.

Маршрут називається ***ланцюгом****, якщо кожне ребро зустрічається в ньому не більше ніж один раз, і* ***простим ланцюгом****, якщо будь-яка вершина (крім, можливо, початкової) зустрічається в ньому не більше, як один раз.*

Якщо ланцюг є замкненим, то його називають ***циклом***, а якщо простий ланцюг − замкнений, то це – простий цикл. Визначення маршруту легко перенести з графа на орієнтований граф. Маршрут в останньому називатимемо ***шляхом***. Відповідно можна перенести також визначення ланцюга, простого ланцюга та циклу. Простий цикл в орієнтованому графі ще називається ***контуром***.

Довжина найменшого ланцюга між вершинами v і w звичайного графа G називається ***відстанню*** $d(v, w)$ *між цими вершинами.* Відстань задовольняє аксіомам метрики: $d(v, w) \geq 0$; $d(v, w) = 0$, якщо $v = w$; $d(v, w) = d(w, v)$;

$d(v, w) + d(w, u) \geq d(v, u)$.

Дві вершини v і w називаються зв'язаними, якщо існує маршрут з кінцями v та w. Граф називається зв'язним, якщо будь-яка пара його вершин є зв'язаною.

Якщо граф не є зв'язним, то він називається незв'язним.

Зв'язністю графа називається мінімальна кількість вершин, вилучення яких приводить до утворення незв'язного графа.

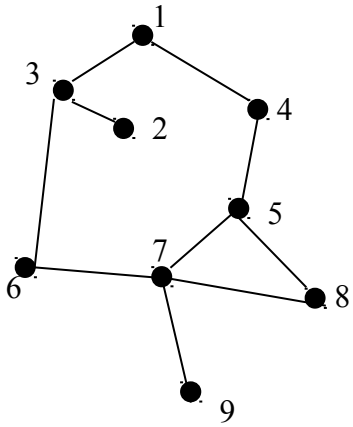
10.2 Алгоритми перегляду вершин графів

Значна кількість алгоритмів на графах вимагає перегляду вершин графа. Існує два класичні алгоритми перегляду всіх вершин графа: пошук у глибину і пошук у ширину.

Пошук у глибину

Пошук починається з деякої фіксованої вершини *v*. Розглядається вершина *u*, суміжна з *v*. Вона вибирається. Процес повторюється з вершиною *u*. Якщо на черговому кроці ми працюємо з вершиною *q* і немає вершин, суміжних з *q* і не розглянутих раніше (нових), то повертаємося з вершини *q* до вершини, яка була до неї. У тому випадку, коли це вершина *v*, процес перегляду закінчений. Для реалізації даного методу потрібна структура даних – стек. Розглянемо приклад пошуку в глибину на графі, описаному матрицею суміжності (рис. 10.7). Почнемо пошук з першої вершини. Нехай:

- Матриця сигналу перегляду чергової вершини – FLAG;
- Стек для зберігання номерів переглянутих вершин – ST;
- Покажчик вершини стека – Y;
- Номер чергової вершини – T.



	1	2	3	4	5	6	7	8	9
1			1	1					
2			1						
3	1	1				1			
4	1				1				
5				1			1	1	
6			1				1		
7					1	1		1	1
8					1		1		
9							1		

Рисунок 10.7 − Приклад графа та його матриці суміжності.

Трасировка пошуку в глибину для даного прикладу буде виглядати, як наведено на рис. 10.8. Черга огляду вершин графа наведена на рис. 10.9

Y	ST	FLAG	T
1	1	0 1 1 1 1 1 1 1 1 1	1
2	3 1	0 1 0 1 1 1 1 1 1 1	3
3	2 3 1	0 0 0 1 1 1 1 1 1 1	2
2	3 1	0 0 0 1 1 1 1 1 1 1	3
3	6 3 1	0 0 0 1 1 0 1 1 1 1	6
4	7 6 3 1	0 0 0 1 1 0 0 1 1 1	7
5	5 7 6 3 1	0 0 0 1 0 0 0 1 1 1	5
6	4 5 7 6 3 1	0 0 0 0 0 0 0 1 1 1	4
5	5 7 6 3 1	0 0 0 0 0 0 0 1 1 1	5
6	8 5 7 6 3 1	0 0 0 0 0 0 0 0 0 1	8
5	5 7 6 3 1	0 0 0 0 0 0 0 0 0 1	5
4	7 6 3 1	0 0 0 0 0 0 0 0 0 1	7
5	9 7 6 3 1	0 0 0 0 0 0 0 0 0 0	9
4	7 6 3 1	0 0 0 0 0 0 0 0 0 0	7
3	6 3 1	0 0 0 0 0 0 0 0 0 0	6
2	3 1	0 0 0 0 0 0 0 0 0 0	3
1	1	0 0 0 0 0 0 0 0 0 0	1
0			

Рисунок 10.8 Трасировка пошуку у глибину для даного приклада

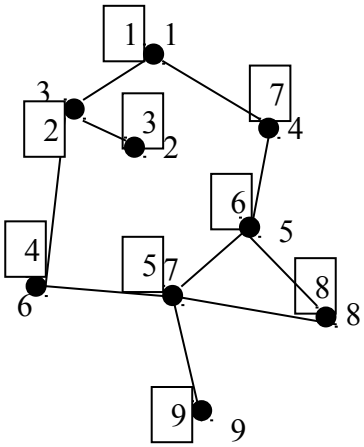
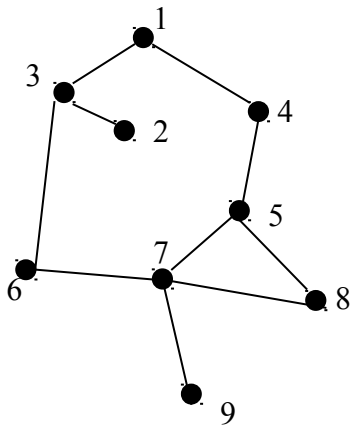


Рисунок 10.9 Черга огляду вершин графа при пошуку у глибину (номер у квадратику)

Пошук у ширин.

Суть методу полягає в тому, щоб розглядати всі вершини, пов'язані з поточною. Принцип вибору наступної вершини – вибирається та, яку було раніше розглянуто. Для реалізації даного принципу необхідна структура даних − черга. Розглянемо приклад пошуку в глибину на графі, описаному матрицею суміжності (рис. 10.10). Почнемо пошук з першої вершини. Нехай:

- Матриця сигналу перегляду чергової вершини – FLAG;
- Черга перегляду вершин – ORDER;
- Покажчик початку черги – X;
- Покажчик кінця черзі – Y.



	1	2	3	4	5	6	7	8	9
1			1	1					
2			1						
3	1	1				1			
4	1				1				
5				1			1	1	
6			1				1		
7					1	1		1	1
8					1		1		
9							1		

Рисунок 10.10 − Приклад графа та його матриці суміжності

Трасировку пошуку в ширину для наведеного прикладу подано на рис. 10.11, а черга огляду графа− на рис. 10.12.

T	ORDER	FLAG
1	3 4	0 1 0 0 1 1 1 1 1
3	4 2 6	0 0 0 0 1 0 1 1 1
4	2 6 5	0 0 0 0 0 0 1 1 1
2	6 5	0 0 0 0 0 0 1 1 1
6	5 7	0 0 0 0 0 0 0 1 1
5	7 8	0 0 0 0 0 0 0 0 1
7	8 9	0 0 0 0 0 0 0 0 0
8	9	0 0 0 0 0 0 0 0 0
9		0 0 0 0 0 0 0 0 0

Рисунок 10.11 – Трасировка пошуку в ширину для даного прикладу

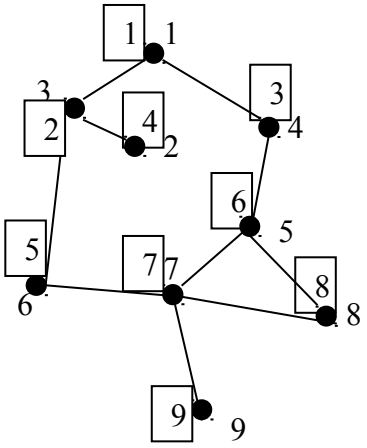


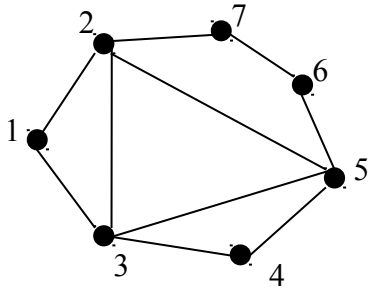
Рисунок 10.12– Черга огляду вершин графа при пошуку в ширину

Ейлерові цикли

Ейлеровий цикл – це такий цикл, який проходить рівно один раз по кожному ребру.

Теорема. *Зв'язаний неорієнтований граф G містить Ейлеровий цикл тоді і тільки тоді, коли число вершин непарного ступеня дорівнює нулю. Ступінь вершини – кількість ребер, інцидентних з цією вершиною.*

Пошук Ейлерого циклу. Дано граф, що задовольняє умові теореми. Потрібно знайти Ейлеровий цикл. Використовується перегляд графа в глибину, при цьому ребра видаляються. Порядок перегляду (номера вершин) запам'ятовуються в першому стеку. При виявленні вершини, з якої не виходять ребра, її номер записується в другий стек. Процес триває до тих пір, поки перший стек не стане порожнім. У другому стеку після цього будуть записані номери вершин графа в порядку відповідному Ейлеровому циклу.



	1	2	3	4	5	6	7
1		1	1				
2	1		1		1		1
3	1	1		1	1		
4			1		1		
5		1	1	1		1	
6					1		1
7		1				1	

Рисунок 10.13 − Приклад графа та його матриці суміжності

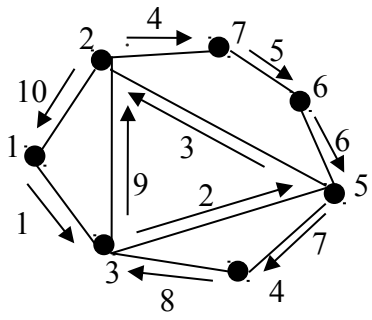


Рисунок 10.14 − Черга огляду вершин графа при пошуку Ейлерового циклу

На рис. 10.13 наведений приклад зв’язаного неорієнтованого графа, у якого число непарних вершин дорівнює нулю. На рис. 10.14 стрілками з цифрами показано Ейлеровий цикл даного графа.

Гамільтоновий цикл

Граф називається Гамільтоновим, якщо в ньому є цикл, що містить кожну вершину цього графа. Сам цикл також називається Гамільтоновим. Не всі пов'язані графи є Гамільтонови.

Розглянемо пошук Гамильтонового циклу на прикладі графа рис. 10. 15. Дано зв'язаний неорієнтований граф. Потрібно знайти усі Гамільтонови цикли графа, якщо вони є. Метод розв’язування − перебір з поверненням. Починаємо пошук розв’язування, наприклад, з першої вершини графа. Припустимо, що вже знайдено перші k компонент розв’язування. Розглядаємо ребра, що виходять з останньої вершини. Якщо є такі, що йдуть в раніше не переглянуті вершини, то включаємо цю вершину в розв’язання і позначаємо як переглянутою. Отримано $k+1$ компонентів розв’язування. Якщо таких компонентів немає, то повертаємося до попередньої вершини і намагаємося знайти ребро з неї, що виходить в іншу вершину. Розв’язування отримано при перегляді всіх вершин графа і можливості досягти з останньої вершини першу. Розв’язування виводиться, і триває процес знаходження наступних циклів.

Рисунок 10.16− Частина дерева, яка

На рис. 10.16 подано механізм пошуку Гамільтонових циклів, які починаються з вершини 1 та закінчуються в цій же вершині, коли переглянуто всі вершини. Процедура находження Гамільтонова циклу є рекурсивною.

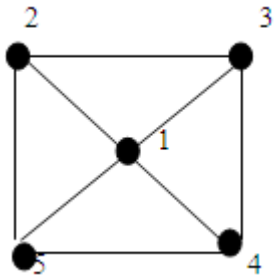


Рисунок 10.15 − Приклад зв'язаного графа для розгляду Гамільтонового циклу

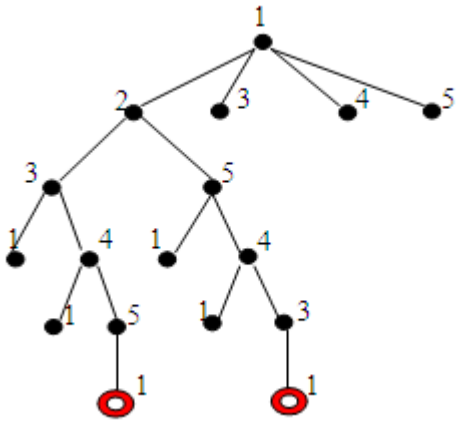


Рисунок 10.16− Частина дерева, яка показує механізм роботи методу

ТЕМА 11. ПРОГРАМУВАННЯ ЗАДАЧ ОПТИМІЗАЦІЇ З ГРАФАМИ. АЛГОРИТМ ФЛОЙДА

11.1 Метод Флойда обчислення найкоротших шляхів у графі

Алгоритм Флойда є одним з методів пошуку найкоротших шляхів у графі. Перш ніж представляти алгоритм, необхідно ввести деякі позначення. Перенумеруємо вершини вихідного графа цілими числами від 1 до N. Позначимо через $d_{i,j}^{m}$ довжину найкоротшого шляху з вершини i у вершину j, який у якості проміжних може містити тільки перші m вершин графа. Нагадаємо, що проміжною вершиною шляху є будь-яка вершина, яка належить йому та не збігається з його початковою або кінцево вершинами. Якщо між вершинами i та j не існує жодного шляху зазначеного типу, то умовно будемо вважати, що $d_{i,j}^{m} = \infty$. З даного визначення величин $d_{i,j}^{m}$ випливає, що величина $d_{i,j}^{0}$, представляє довжину найкоротшого шляху з вершини i у вершину j, що не має проміжних вершин, тобто довжину найкоротшої дуги, що з'єднує i з j (якщо такі дуги присутні в графі). Для будь-якої вершини i покладемо $d_{i,i}^{m} = 0$. Відзначимо далі, що величина $d_{i,j}^{m}$ представляє довжину найкоротшого шляху між вершинами i та j .

Позначимо через D^m матрицю розміру NxN, елемент (i, j) якої збігається з $d_{i,j}^{m}$. Якщо у вихідному графі нам відома довжина кожної дуги, то ми можемо сформувати матрицю D^0. Наша мета полягає у визначенні матриці D^N, що представляє найкоротші шляхи між усіма вершинами розглянутого графа.

В алгоритмі Флойда в якості вихідної виступає матриця D^0. Спочатку з цієї матриці обчислюється матриця D^1. Потім за матрицею D^1 обчислюється матриця D^2 і т. д. Процес повторюється до тих пір, доки за матрицею D^{N-1} не буде обчислена матриця D^N.

Розглянемо головну ідею, що лежить в основі алгоритму Флойда. Суть алгоритму Флойда полягає у перевірці того, чи не виявиться шлях з вершини i у вершину j коротше, якщо він буде проходити через деяку проміжну вершину m. Припустимо, що нам відомі:

1) найкоротший шлях з вершини i у вершину m, в якому в якості проміжних допускається використання тільки перших $(m - 1)$ вершин;

2) найкоротший шлях з вершини m у вершину j, в якому в якості проміжних допускається використання тільки перших $(m - 1)$ вершин;

3) найкоротший шлях з вершини i у вершину j, в якому в якості проміжних допускається використання тільки перших $(m - 1)$ вершин.

Оскільки, за припущенням, вихідний граф не може містити контурів негативної довжини, один з двох шляхів − шлях, що співпадає з представленим у пункті 3, або шлях, який є об'єднанням шляхів з пунктів 1 і 2 - повинен бути найкоротшим шляхом з вершини i у вершину j, в якому в якості проміжних допускається використання тільки перших m вершин. Таким чином, $d_{i,j}^{m} = \min \{d_{i,m}^{m-1} + d_{m,j}^{m-1}; d_{i,j}^{m-1}\}$.

Зі співвідношення видно, що для обчислення елементів матриці D^m необхідно мати лише елементи матриці D^{m-1}. Більше того, відповідні

обчислення можуть бути проведені без звернення до вихідного графа. Тепер ми можем дати формальний опис алгоритму Флойда для знаходження на графі найкоротших шляхів між усіма парами вершин.

Алгоритм Флойда

1. Пронумерувати вершини графа від 1 до N цілими числами, визначити матрицю D^0, кожен елемент $d_{i,j}$ якої є довжина найкоротшої дуги між вершинами i та j. Якщо такої дуги немає, покласти значення елемента рівним ∞. Крім того, покласти значення діагонального елемента $d_{i,i}$ рівним 0.

2. Для цілого m, яке послідовно набуває значення 1 ... N, визначити за елементами матриці D^{m-1} елементи D^m.

3. Алгоритм закінчується отриманням матриці всіх найкоротших шляхів D^N, N − число вершин графа.

11.2 Приклад визначення найкоротшого шляху за допомогою алгоритму Флойда

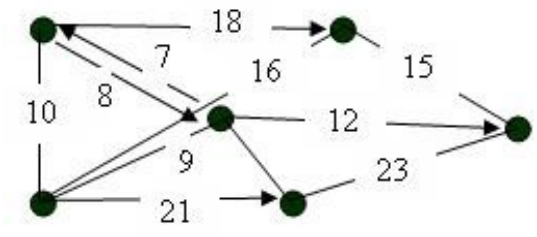


Рисунок 11.1− Орієнтований граф

Необхідно знайти найкоротший шлях між кожною парою вершин у графі, який представлено на рис. 11.1. Пронумеруємо усі вершини графа, та складемо матрицю довжини найкоротших дуг D^0, у випадку, якщо дуги поміж вершиною i та j не існує, елементу $d_{i,j}$ матриці надається значення ∞. Вихідний граф з пронумерованими вершинами зображений на рис. 11.2.

Матриця D^0:

D^0=	0	10	18	8	∞	∞
	10	0	16	9	21	∞
	∞	16	0	∞	∞	15
	7	9	∞	0	∞	12
	∞	∞	∞	∞	0	23
	∞	∞	15	∞	23	0

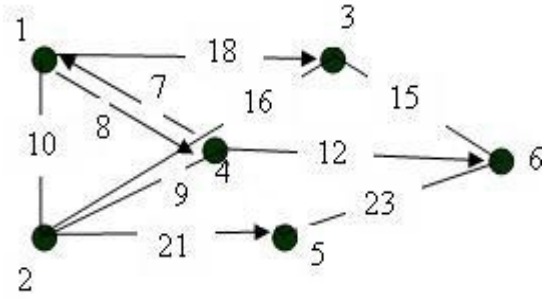


Рисунок 11.2− Вихідний граф з пронумерованими вершинами

На підставі матриці D^0, розраховуємо послідовно усі елементи матриці D^1. Для цього ми використовуємо рекурентне співвідношення

$$d_{i,j}^{1}=\min\{ d_{i,1}^{0}+ d_{1,j}^{0}; d_{i,j}^{0}\}$$
$$d_{1,1}^{1}= \min\{d_{1,1}^{0}+d_{1,1}^{0}d_{1,1}^{0}\}= \min\{0+0; 0\}=0$$
$$d_{1,2}^{1}= \min\{d_{1,1}^{0}+d_{1,2}^{0}d_{1,2}^{0}\}= \min\{0+10; 10\}=10$$
$$d_{1,3}^{1}= \min\{d_{1,1}^{0}+d_{1,3}^{0}d_{1,3}^{0}\}= \min\{0+18; 18\}=18$$

$d_{1,4}^1 = \min\{d_{1,1}^0 + d_{1,4}^0 d_{1,4}^0\} = \min\{0+8;\ 8\} = 8$
$d_{1,5}^1 = \min\{d_{1,1}^0 + d_{1,5}^0 d_{1,5}^0\} = \min\{0+\infty;\ \infty\} = \infty$
$d_{1,6}^1 = \min\{d_{1,1}^0 + d_{1,6}^0 d_{1,6}^0\} = \min\{0+\infty;\ \infty\} = \infty$
$d_{2,1}^1 = \min\{d_{2,1}^0 + d_{1,1}^0 d_{2,1}^0\} = \min\{10+0;\ 10\} = 10$
$d_{2,2}^1 = \min\{d_{2,1}^0 + d_{1,2}^0 d_{2,2}^0\} = \min\{10+10;\ 0\} = 0$
$d_{2,3}^1 = \min\{d_{2,1}^0 + d_{1,3}^0 d_{2,3}^0\} = \min\{10+18;\ 16\} = 16$
$d_{2,4}^1 = \min\{d_{2,1}^0 + d_{1,4}^0 d_{2,4}^0\} = \min\{10+8;\ 9\} = 9$
$d_{2,5}^1 = \min\{d_{2,1}^0 + d_{1,5}^0 d_{2,5}^0\} = \min\{10+\infty;\ 21\} = 21$
$d_{2,6}^1 = \min\{d_{2,1}^0 + d_{1,6}^0 d_{2,6}^0\} = \min\{10+\infty;\ \infty\} = \infty$
$d_{3,1}^1 = \min\{d_{3,1}^0 + d_{1,1}^0 d_{3,1}^0\} = \min\{\infty+0;\ \infty\} = \infty$
$d_{3,2}^1 = \min\{d_{3,1}^0 + d_{1,2}^0 d_{3,2}^0\} = \min\{\infty+10;\ 16\} = 16$
$d_{3,3}^1 = \min\{d_{3,1}^0 + d_{1,3}^0 d_{3,3}^0\} = \min\{\infty+18;\ 0\} = 0$
$d_{3,4}^1 = \min\{d_{3,1}^0 + d_{1,4}^0 d_{3,4}^0\} = \min\{\infty+8;\ \infty\} = \infty$
$d_{3,5}^1 = \min\{d_{3,1}^0 + d_{1,5}^0 d_{3,5}^0\} = \min\{\infty+\infty;\ \infty\} = \infty$
$d_{3,6}^1 = \min\{d_{3,1}^0 + d_{1,6}^0 d_{3,6}^0\} = \min\{\infty+\infty;\ 15\} = 15$
$d_{4,1}^1 = \min\{d_{4,1}^0 + d_{1,1}^0 d_{4,1}^0\} = \min\{7+0;\ 7\} = 7$
$d_{4,2}^1 = \min\{d_{4,1}^0 + d_{1,2}^0 d_{4,2}^0\} = \min\{7+10;\ 9\} = 9$
$d_{4,3}^1 = \min\{d_{4,1}^0 + d_{1,3}^0 d_{4,3}^0\} = \min\{7+18;\ \infty\} = 25$
$d_{4,4}^1 = \min\{d_{4,1}^0 + d_{1,4}^0 d_{4,4}^0\} = \min\{7+8;\ 0\} = 0$
$d_{4,5}^1 = \min\{d_{4,1}^0 + d_{1,5}^0 d_{4,5}^0\} = \min\{7+\infty;\ \infty\} = \infty$
$d_{4,6}^1 = \min\{d_{4,1}^0 + d_{1,6}^0 d_{4,6}^0\} = \min\{7+\infty;\ 12\} = 12$
$d_{5,1}^1 = \min\{d_{5,1}^0 + d_{1,1}^0 d_{5,1}^0\} = \min\{\infty+0;\ \infty\} = \infty$
$d_{5,2}^1 = \min\{d_{5,1}^0 + d_{1,2}^0 d_{5,2}^0\} = \min\{\infty+10;\ \infty\} = \infty$
$d_{5,3}^1 = \min\{d_{5,1}^0 + d_{1,3}^0 d_{5,3}^0\} = \min\{\infty+18;\ \infty\} = \infty$
$d_{5,4}^1 = \min\{d_{5,1}^0 + d_{1,4}^0 d_{5,4}^0\} = \min\{\infty+8;\ \infty\} = \infty$
$d_{5,5}^1 = \min\{d_{5,1}^0 + d_{1,5}^0 d_{5,5}^0\} = \min\{\infty+\infty;\ 0\} = 0$
$d_{5,6}^1 = \min\{d_{5,1}^0 + d_{1,6}^0 d_{5,6}^0\} = \min\{\infty+\infty;\ 23\} = 23$
$d_{6,1}^1 = \min\{d_{6,1}^0 + d_{1,1}^0 d_{6,1}^0\} = \min\{\infty+0;\ \infty\} = \infty$
$d_{6,2}^1 = \min\{d_{6,1}^0 + d_{1,2}^0 d_{6,2}^0\} = \min\{\infty+10;\ \infty\} = \infty$
$d_{6,3}^1 = \min\{d_{6,1}^0 + d_{1,3}^0 d_{6,3}^0\} = \min\{\infty+18;\ 15\} = 15$
$d_{6,4}^1 = \min\{d_{6,1}^0 + d_{1,4}^0 d_{6,4}^0\} = \min\{\infty+8;\ \infty\} = \infty$
$d_{6,5}^1 = \min\{d_{6,1}^0 + d_{1,5}^0 d_{6,5}^0\} = \min\{\infty+\infty;\ 23\} = 23$
$d_{6,6}^1 = \min\{d_{6,1}^0 + d_{1,6}^0 d_{6,6}^0\} = \min\{\infty+\infty;\ 0\} = 0$

Запишемо матрицю D^1, включивши в неї розраховані елементи.

$D^1 =$

	0	10	18	8	∞	∞
	10	0	16	9	21	∞
	∞	16	0	∞	∞	15
	7	9	25	0	∞	12
	∞	∞	∞	∞	0	23
	∞	∞	15	∞	23	0

На основі матриці D^1, аналогічно обчислимо послідовно усі елементи матриці D^2. Для цього ми використовуємо рекурентне співвідношення:

$d_{i,j}^{\,2} = \min\{ d_{i,2}^{\,1} + d_{2,j}^{\,1};\ d_{i,j}^{\,1}\}$.

$D^2=$

0	10	18	8	31	∞
10	0	16	9	21	∞
26	16	0	25	37	15
7	9	25	0	30	12
∞	∞	∞	∞	0	23
∞	∞	15	∞	23	0

На основі матриці D^2, обчислимо послідовно усі елементи матриці D^3, а далі матриці D^4, D^5, D^6.

$D^3=$

0	10	18	8	31	33
10	0	16	9	21	31
26	16	0	25	37	15
7	9	25	0	30	12
∞	∞	∞	∞	0	23
41	31	15	40	23	0

$D^4=$

0	10	18	8	31	20
10	0	16	9	21	21
26	16	0	25	37	15
7	9	25	0	30	12
∞	∞	∞	∞	0	23
41	31	15	40	23	0

$D^5=$

0	10	18	8	31	20
10	0	16	9	21	21
26	16	0	25	37	15
7	9	25	0	30	12
∞	∞	∞	∞	0	23
41	31	15	40	23	0

$D^6=$

0	10	18	8	31	20
10	0	16	9	21	21
26	16	0	25	37	15
7	9	25	0	30	12
64	54	38	63	0	23
41	31	15	40	23	0

Отже, ми отримали матрицю довжин найкоротших шляхів між кожною парою вершин графа. Нижче подано таблицю шляхів. Кожний елемент d_{i-j} таблиці, це шлях з вершини i до вершини j:

Таблиця шляхів

i*/*j	**1**	**2**	**3**	**4**	**5**	**6**
1	-	d_{1-2}=1-2	d_{1-3}=1-3	d_{1-4}=1-4	d_{1-5}=1-2-5	d_{1-6}=1-4-6
2	d_{2-1}=2-1	-	d_{2-3}=2-3	d_{2-4}=2-4	d_{2-5}=2-5	d_{2-6}=2-4-6
3	d_{3-1}=3-2-1	d_{3-2}=3-2	-	d_{3-4}=3-2-4	d_{3-5}=3-2-5	d_{3-6}=3-6

4	$d_{4\text{-}1}$=4-1	$d_{4\text{-}2}$=4-2	$d_{4\text{-}3}$=4-1-3	-	$d_{4\text{-}5}$=4-2-5	$d_{4\text{-}6}$=4-6
5	$d_{5\text{-}1}$=5-6-3-2-1	$d_{5\text{-}2}$=5-6-3-2	$d_{5\text{-}3}$=5-6-3	$d_{5\text{-}4}$=5-6-3-2-4	-	$d_{5\text{-}6}$=5-6
6	$d_{6\text{-}1}$=6-3-2-1	$d_{6\text{-}2}$=6-3-2	$d_{6\text{-}3}$=6-3	$d_{6\text{-}4}$=6-3-2-4	$d_{6\text{-}5}$=6-5	-

ПЕРЕЛІК ЗНАНЬ ТА УМІНЬ, ЯКИХ МАЄ НАБУТИ СТУДЕНТ У ПРОЦЕСІ ВИВЧЕННЯ МАТЕРІАЛУ МОДУЛЯ 2

Знання:

- *обчислювальні можливості пакета MatLab;*
- *методи розв'язування диференціальних рівнянь;*
- *апроксимація та інтерполяція функції;*
- *чисельні методи одновимірної оптимізації;*
- *методи оптимізації багатовимірних функцій;*
- *багатовимірні функції з обмеженнями, задачі лінійного програмування;*
- *чисельні методи розв'язування задач лінійного програмування;*
- *алгоритми перегляду вершин графа;*
- *алгоритм визначення найкоротшого шляху перевезень за допомогою графа .*

Уміння:

- *обчислювати функції та будувати графіки у MatLab;*
- *розв'язувати диференціальні рівняння у MathCad та C++;*
- *визначати області унімодальності й обчислювати максимальне або мінімальне значення одновимірної функції у MathCad та C++;*
- *розв'язувати задачі лінійного програмування у у MathCad та Excel; обчислювати найкоротші шляхи перевезення у C++.*

ЗАВДАННЯ -ТЕСТИ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ ТА УМІНЬ

1. *Запишіть номер правильної, на Ваш погляд, відповіді.*
   Задача Коші при розв'язанні диференціальних рівнянь дозволяє:
   1 *визначити область допустимих розв'язань;*
   2 *визначити умови збіжності розв'язання;*
   3 *виділити єдине розв'язання з множини розв'язань диференціального рівняння;*
   4 *отримати умову виходу з процесу наближення;*
   5 *перетворити диференціальне рівняння n порядку у систему n диференціальних рівнянь першого порядку.*
2. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
   Методи Рунге – Кути мають такі властивості:
   1 *одноступінчастими: щоб обчислити y_{m+1}, потрібна інформація про попередню точку (x_m, y_m);*
   2 *перша похідна y(x) змінює знак на проміжку [a,b];*
   3 *збігаються з методом Тейлора аж до членів порядку h^P;*
   4 *додаткові умови задаються у вигляді функціональних співвідношень між шуканими розв'язками;*
   5 *не вимагають обчислення похідних від f(x, y), а вимагають обчислення самої функції.*
3. *Запишіть номер правильної, на Ваш погляд, відповіді.*
   Метод Ейлера дозволяє знайти розв'язання:
   1 *системи лінійних рівнянь;*
   2 *нелінійного рівняння;*
   3 *задачі отимізації функції однієї змінної;*
   4 *задачі лінійного програмування;*
   5 *диференціального рівняння.*
4. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
   Міра близькості вихідної функції f(x) та наближеної φ(x) у задачах апроксимації визначається умовами:
   1 $\max|f(x_i) - \varphi(x_i)| < \varepsilon$;
   2 $\min \sum_{i=1}^{n} |f(x_i - \varphi(x_i)|$;
   3 $\Delta = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n} \left(f(x_i) - \varphi(x_i)\right)^2} = \min.$;
   4 $f(x_i) = \varphi(x_i)$;
   5 $|f(x_i) - \varphi(x_i)| > 0$.
5. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
   Метод найменших квадратів це:
   1 *апроксимація функції F_n поліномом степеня m ($m<n$);*
   2 *інтерполяція алгебраїчним многочленом;*
   3 *апроксимація за умовою рівномірного наближення ;*

4 *апроксимація за умовою* $\Delta = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n}\left(f(x_i) - \varphi(x_i)\right)^2} = \min$;

5 *апроксимація поліномом другого степеня.*

6. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
   Інтерполяція це:
   1 *побудова такої функції φ(x), яка приблизно (у певному сенсі) дорівнює вихідній функції f(x) на розглянутому відрізку [a,b];*
   2 *спосіб, за допомогою якого за таблицею, що містить певні числові дані, можна знайти проміжні результати, яких нема безпосередньо в таблиці;*
   3 *наближення за умовою* $f(x_i) = \varphi(x_i)$;
   4 *коли кожне нове уточнене наближення розв'язання* x_i *обчислюється через попереднє* x_{i-1};
   5 *наближення за умовою* $\max|f(x_i) - \varphi(x_i)| < \varepsilon$.
7. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
   До методів одновимірної оптимізації належать:
   1 *метод трапеції;*
   2 *метод бісекції;*
   3 *метод Ньютона;*
   4 *метод рівномірного пошуку;*
   5 *метод хорд.*
8. *Запишіть номер правильної, на Ваш погляд, відповіді:*
   Формула $y_{m+1} = y_m + h*f(x_m, y_m)$ *відповідає формулі обчислення:*
   1 *методом Ньютона;*
   2 *методом прямокутників;*
   3 *методом Ейлера;*
   4 *методом ітерації.*
9. *Запишіть номер правильної, на Ваш погляд, відповіді.*
   Метод одновимірної оптимізації, у якому інтервал унімодальності ділиться навпіл, це:
   1 *метод рівномірного пошуку;*
   2 *метод «золотого перетину»;*
   3 *метод бісекції;*
   4 *метод Ньютона.*
10. *Запишіть номер правильної, на Ваш погляд, відповіді.*
    Умова, яка перевіряється при пошуку максимума (мінімума), це:
    1 $|x_i - x_{i-1}| < \varepsilon$;
    2 $f(a)\cdot f(b) < 0$;
    3 $F1 < F2$;
    4 $f(a)\cdot f^{1}(a) > 0$;
    5 $|a - b| < \varepsilon$.
11. *Перелічіть номери правильних, на Ваш погляд, відповідей:*
    Методи багатовимірної оптимізації це:
    1 *метод Зейделя;*

2 метод градієнтного спуску;
3 метод «золотого перетину»;
4 симплекс-метод;
5 метод Ньютона.

12. Запишіть номер правильної, на Ваш погляд, відповіді.
Ідея методу градієнтного спуску полягає :
1 в оптимізації функції вздовж кожної з координат;
2 в розрахунку напрямку з координатами $\{\frac{\partial \overline{F}}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\}$ *;*
3 розв'язання системи рівнянь $\frac{\partial F}{\partial x} = 0; \frac{\partial F}{\partial y} = 0; \frac{\partial F}{\partial z} = 0$

13. Запишіть номер правильної, на Ваш погляд, відповіді.
Задача лінійного програмування це:
1 коли цільова функція та функції обмеження лінійні;
2 коли багатовимірна функція – лінійна;
3 цільова функція та функції обмеження мають вигляд $F(X) = \min$;
$\varphi_i(X)=0; \quad 1< i < m; \qquad \psi_j(X) \geq 0; \qquad m < j < p.$

14. Перелічіть номери правильних, на Ваш погляд, відповідей:
Методи розв'язування задач лінійного програмування – це:
1 метод координатного спуску;
2 графічний метод;
3 метод градієнтного спуску;
4 симплекс-метод.

15. Запишіть номер правильної, на Ваш погляд, відповіді.
Ідея симплекс-методу полягає у наступному:
1 у переборі усіх вершин області припустимих розв'язань;
2 вибір вершин в області припустимих розв'язань, які знаходяться в напрямку максимуму (мінімуму) багатовимірної функції;
3 оптимізація багатовимірної функції вздовж кожної координати.

16. Запишіть номер правильної, на Ваш погляд, відповіді.
Цільова функція та функції обмеження транспортної задачі записуються так:
1 $F(x) = \Sigma c_i x_i = \min, \quad x_i \geq 0, 1 \leq i \leq n, \quad \Sigma a_{ji} x_i = b_j, 1 \leq j \leq m,$ $\Sigma a_{ji} x_i \leq b_j, m \leq j \leq M$ *(сума по і від 1 до n) .*
2 $F(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} = \min ; \sum_{j=1}^{n} x_{ij} = a_i ; \quad \sum_{i=1}^{m} x_{ij} = b_j ; \quad x_{ij} \geq 0, 1 \leq i \leq m, 1 \leq j \leq n.$
3 $F(x) = \Sigma c_i x_i = \min, x_i \geq 0, 1 \leq i \leq N, \Sigma a_{ji} x_i = b_j, 1 \leq j \leq M (M < N).$
4 $F(x) = \min, \quad \varphi_i(x) = 0, 1 \leq i \leq m, \quad \psi_j(x) \geq 0, 1 \leq j \leq p .$

17. Запишіть номер правильної, на Ваш погляд, відповіді.
Графічний метод розв'язання задачі ЛП полягає у наступному:
1 Побудова графіків функцій обмеження, обчислення координат вершин області припустимих розв'язань та визначення оптимального з них;

2 *Побудова графіків функцій обмеження, за допомогою зсуву цільової функції у напрямку найскорішого зростання (зменшення) визначити оптимальну вершину області припустимих рішень;*

3 *За допомогою функцій обмеження на графіку визначити область припустимих розв'язань та знайти точки перетинання цільової функції з цією областю.*

18. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
Цільова функція та функції обмеження транспортної задачі записуються так:

1 $F(x) = \Sigma c_i x_i = min,\ x_i \geq 0,\ 1 \leq i \leq n;$

2 $\sum_{j=1}^{n} x_{ij} = a_i;\quad \sum_{i=1}^{m} x_{ij} = b_j;$

3 $x_i \geq 0,\ 1 \leq i \leq N,\ \Sigma a_{ji} x_i = b_j,\ 1 \leq j \leq M\ (M < N);$

4 $\varphi_i(x) = 0,\ 1 \leq i \leq m,\quad \psi_j(x) \geq 0,\ 1 \leq j \leq p\ .$

5 $F(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} = \min.$

19. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
Граф можна описати за допомогою:

1 *транспонованої матриці;*

2 *матриці інцидентності;*

3 *матриці суміжності;*

4 *матриці довжини найкоротших дуг;*

5 *матриці вершин графа.*

20. *Запишіть номер правильної, на Ваш погляд, відповіді.*
Ідея алгоритму Флойда полягає у наступному:

1 *у перевірці того, чи не виявиться шлях з вершини і у вершину j коротше, якщо він буде проходити через деяку проміжну вершину m;*

2 *обчисленні довжини усіх дуг, які зв'язують вершини і та j;*

3 *мінімізації функції довжини шляху з вершини і у вершину j.*

21. *Перелічіть номери правильних, на Ваш погляд, відповідей.*
Алгоритм Флойда складається з наступних кроків:

1 *пронумерувати вершини графа, визначити матрицю D^0, кожен елемент $d_{i,j}$ якої є довжина найкоротшої дуги між вершинами і та j;*

2 *для цілого m, яке послідовно приймає значення 1 ... N, визначити за елементами матриці D^{m-1} елементи D^m;*

3 *порівняти елементи матриці D^{m-1} та D^m, визначити мінімальні;*

4 *отримати матрицю всіх найкоротших шляхів D^N, N - число вершин графа;*

5 *отримати матрицю передостанніх вершин у найкоротшому шляху з вершини і у вершину j.*

ЛІТЕРАТУРА

1 **Шаповаленко В. А.** Чисельні методи та моделювання на ЕОМ: Навчальний посібник. Модуль 1. / В. А. Шаповаленко, Л. М. Буката, О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2010. – Ч. 1. – 95 с.

2 **Буката Л.М.** Чисельні методи моделювання об'єктів: Метод. вказівки для лаб. та практ. занять. Модуль 1. / Л. М. Буката, В. А. Шаповаленко, О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2010. – Ч. 2. – 85 с.

3 **Дьяконов В. П.** Mathcad 8 PRO в математике, физике и Internet / В. П. Дьяконов, Н. В. Авраменко. – М. : Нолидж, 1999. – 512 с.

4 **Єщенко А. І.** Основи програмування в математичному пакеті Mathcad / А. І. Єщенко, І. А. Єщенко. – Одеса: УДАЗ, 2000. – 285 с.

5 **Краскевич В. Е.** Численные методы в инженерных исследованиях / В. Е. Краскевич, К. Х. Зеленский, В. И. Гречко. − К. : Вища школа, 1986. – 263 с.

6 **Крячков А. В.** Программирование на С++. Практикум: учеб. пособие для вузов / А. В. Крячков, И. В. Сухинина, В. К. Томшин. − М. : Горячая линия – Телеком, 2000. – 344 с.

7 **Леонов Ю. Г.** Программирование инженерных задач: метод. пособие / Ю. Г. Леонов, Н. В. Силкина, О. Д. Шпинова. – Одесса : ОНАС, 2002. – 68 с.

8 **Поддубный Г. В.** Численные методы и их применение в инженерных расчетах: учебн. пособие / Г. В. Поддубный, Л. И. Соколов. – Одесса : ОЭИС, 1981. – Ч. 1. – 85 с.

9 **Поддубный Г. В.** Численные методы и их применение в инженерных расчетах: учебн. Пособ. / Г. В. Поддубный, Л. И. Соколов. – Одесса : ОЭИС, 1983. – Ч. 2. – 85 с.

10 **Фельдман Л. П.** Чисельні методи в інформатиці / Л. П. Фельдман, А. І. Петренко, О. А. Дмитрієва. – К. : BHV, 2006. – 480 с.

11 **Мэтьюз Д.Г.** Численные методы. Использование MATLAB; пер. с англ. / Д. Г. Мэтьюз, К. Д. Финк. – М. : Вильямс, 2001. – 720 с.

12 **Дьяконов В. П**. MatLab 6/6.1/6.5+Simulink 4/5. Основы применения – М. : Солон-Пресс, 2002. – 768 с.

13 **Калиткин Н.Н**. Численные методы.− М.:Наука, 1978

14 **Поповський В.В**. Математичні основи теорії телекомунікаційних систем; Под. ред. В.В. Поповського., - Харків «Компанія СМІТ», 2006.

15 **Щуровская А.Ю.** Математическое программирование. МОДУЛЬ №1−Часть 1: учеб. пособ.− Одесса: ОНАС,2008.

ЗМІСТ

Здано до набору 18.10.13. Підписано до друку 2.11.13.
Обсяг 3,05 ум.-друк. арк.
Формат 90х60/16. Зам. № 5 . Наклад 50 прим.
Віддруковано на видавничому обладнанні фірми RISO
в друкарні редакційно-видавничого центру ОНАЗ ім. О.С. Попова
Одеса, 65021, вул. Ковалевського, 5
Тел. (0482) 705-04-94