Міністерство освіти і науки України

Київський національний університет технологій та дизайну

Кафедра комп'ютерних наук

МЕТОДИ ПРЕДСТАВЛЕННЯ, ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

ЩЕРБАНЬ В.Ю.
КРАСНИТСЬКИЙ С.М.
АСТІСТОВА Т.І.
ЯХНО В.М.

МЕТОДИ ПРЕДСТАВЛЕННЯ, ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЩЕРБАНЬ В.Ю., КРАСНИТСЬКИЙ С.М., АСТІСТОВА Т.І., ЯХНО В.М.

МЕТОДИ ПРЕДСТАВЛЕННЯ, ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Київ – 2023

УДК 004.42
ББК 65.9(4Укр)306.4-6
Щ 610

Рекомендовано Вченою радою Київського національного університету технологій та дизайну для широкого кола викладачів, науковців, аспірантів, магістрів та студентів профільних вищих навчальних закладів що навчаються за освітньою програмою 122 Комп'ютерні науки, інженерно-технічних працівників комп'ютерної галузі (Протокол №11 від 30 червня 2023 року)

Колектив авторів:

ЩЕРБАНЬ В. Ю. – лауреат Державної премії України в галузі науки і техніки, академік Міжнародної академії комп'ютерних наук та систем, доктор технічних наук, професор, завідувач кафедрою комп'ютерних наук Київського національного університету технологій та дизайну;

КРАСНИТСЬКИЙ С.М. - доктор фізико-математичних наук, професор кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

АСТІСТОВА Т.І. - кандидат технічних наук, доцент кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

ЯХНО В.М. - кандидат технічних наук, доцент кафедри комп'ютерних наук Київського національного університету технологій та дизайну.

Рецензенти:

ОПАНАСЕНКО В.М. – лауреат Державної премії України в галузі науки і техніки,, д.т.н., професор, провідний науковий співробітник, Інститут кібернетики НАН України;

ЗАЙЦЕВ С.В. – д.т.н., професор, кафедра інформаційних та комп'ютерних систем, Чернігівський національний технологічний університет;

ЩУЦЬКА Г. В. – д.т.н., директор коледжу, Київський фаховий коледж прикладних наук.

Щ 610 Щербань В.Ю. Методи представлення, збереження та аналізу даних інформаційних систем / В.Ю. Щербань, С.М. Краснитський, Т.І. Астістова, В.М. Яхно. – К.: ТОВ "Фастбінд Україна", 2023. – 472 с.

ISBN 978-617-8237-41-7

Наведені результати досліджень з питань представлення, збереження та аналізу даних інформаційних систем, математичного моделювання систем і технологічних процесів, зокрема з використанням математичних моделей, що реалізовуються з використанням чисельних методів, представлення даних моделями регресійного та дисперсійного аналізів, елементів дослідження операцій, формулювання задач лінійного програмування, еквівалентних перетворень в задачах лінійного програмування, описані складові Web–технологій та наведені приклади їх практичної реалізації. Призначена для широкого кола викладачів, науковців, аспірантів, магістрів та студентів спеціальності 122 – комп'ютерні науки, профільних вищих навчальних закладів, інженерно-технічних працівників комп'ютерної галузі

УДК 004.42
ББК 65.9(4Укр)306.4-6

ISBN 978-617-8237-41-7

ЗМІСТ

ВСТУП

За останні десятиріччя інформаційні технології зазнали такого глобального поширення, що зараз уже важко уявити життя сучасної людини без них. На сучасному етапі можна без особливих труднощів навести приклади використання інформаційних технологій у всі галузях: від освіти і до менеджменту. Сьогодні успіх буде мати та фірма, той заклад, який володіє найсучаснішими комп'ютерними технологіями. Значного прогресу можна досягти і в галузі освіти з впровадженням відповідних інформаційних комп'ютерних технологій, які зможуть зробити процес здобуття освіти більш гнучким, індивідуалізованим і одночасно дадуть змогу студентам використовувати глобальні ресурси для навчання, спілкуватись та обмінюватись досвідом.

Технічними засобами виробництва інформації будуть апаратне, програмне і математичне забезпечення цього процесу. З їхньою допомогою відбувається переробка первинної інформації в інформацію нової якості. Виділимо окремо з цих засобів програмні продукти і назвемо їх інструментарієм, а для більшої чіткості можна його конкретизувати, назвавши програмним інструментарієм інформаційної технології.

Інформаційна технологія опрацювання даних використовується для розв'язання добре структурованих задач, стосовно яких є необхідні вхідні дані і відомі алгоритми та інші стандартні процедури їх опрацювання. Ця технологія застосовується на рівні операційної (виконавчої) діяльності персоналу невисокої кваліфікації з метою автоматизації деяких рутинних постійно повторюваних операцій управлінської праці. Тому впровадження інформаційних технологій і систем на цьому рівні істотно підвищить продуктивність праці персоналу, звільнить його від рутинних операцій, можливо, навіть призведе до необхідності скорочення чисельності працівників. Багато даних на рівні операційної діяльності необхідно

зберігати для наступного використання або на цьому ж рівні, або на іншому. Для їхнього збереження створюються бази даних.

Одним із засобів керування розвитком інтелекту і підвищення його організованості на сучасному етапі є інформатизація суспільства, що ґрунтується насамперед на розвитку інформаційних комп'ютерних технологій. Значення інформаційної технології величезне - вона формує передній край науково-технічного прогресу, створює інформаційний фундамент розвитку науки і всіх інших технологій. Головними, визначальними стимулами розвитку інформаційної технології, є соціально-економічні потреби суспільства, і саме зараз суспільство як ніколи зацікавлене в якомога швидшій інформатизації та комп'ютеризації всіх без винятку сфер діяльності. Дуже важливою властивістю інформаційної технології є те, що для неї інформація є не тільки продуктом, але і вихідною сировиною. Особлива роль приділяється всьому комплексу інформаційної технології і техніки в структурній перебудові економіки убік наукоємності. Більш того, інформаційна технологія є свого роду перетворювачем всіх інших галузей господарства, як виробничих, так і невиробничих, основним засобом їхньої автоматизації, якісної зміни продукції і, як наслідок, їх переходу частково або цілком у категорію наукомістких. Пов'язаний з цим і працезаощаджувальний характер інформаційної технології, що реалізується, зокрема, у керуванні багатьма видами робіт і технологічних операцій. Безсумнівною перевагою інформаційної технології є те, що вона сама створює засоби для своєї еволюції. Формування системи, що само розвивається - найважливіший підсумок, досягнутий у сфері інформаційної технології. Таким чином, усі вищевикладені риси інформаційної технології вказують на те, що вона й у майбутньому залишиться самим перспективним видом технології, що допомагає людині впевнено крокувати шляхом прогресу.

1. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СИСТЕМ І ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

Збільшення випуску продукції підприємствами легкої і текстильної промисловості, підвищення її якості безпосередньо залежить від машинобудівників, технологів і конструкторів, які створюють нове високопродуктивне устаткування, удосконалюють ті, що існують і розробляють нові технологічні процеси.

Вирішення вказаних завдань вимагає проведення великого об'єму обчислювальних операцій, тому розробка систем автоматизованого проектування (САПР) є найважливішим завданням і чинником, що багато в чому визначає технічний рівень, якість і ефективність сучасної техніки і технологічних процесів легкої і текстильної промисловості. Застосування математичних методів і ЕОМ істотно скорочує терміни розробки і впровадження нової техніки і технологічних процесів, дозволяє економити матеріали і отримувати оптимальну конструкцію машини (наприклад, як критерії оптимізації можуть бути вибрані мінімальні габаритні розміри, маса, приведений момент інерції, максимальна величина ККД) або оптимальний технологічний процес (тут як критерії оптимізації можуть бути вибрані наступні: мінімальний сумарний час протікання технологічного процесу, мінімальні енерговитрати, мінімізація відходів, збільшення випуску продукції і ін.). Крім того, автоматизація процесу проектування, із застосуванням ЕОМ і пакетів прикладних програм, дозволяє виключити грубі помилки при виконанні розрахунків, які можуть приводити до порушення стійкості (статичною і динамічною) при роботі технологічного устаткування і при протіканні технологічних процесів.

1.1. Математичні моделі і обчислювальний експеримент

Математичною моделлю називається наближений опис основних закономірностей якого-небудь класу об'єктів, систем або процесів за допомогою математичної символіки у вигляді рівнянь, нерівності і ін.

Поява ЕОМ привела до подальшого розвитку методу математичного моделювання для дослідження поведінки складних технічних систем. У загальному випадку процес отримання математичної моделі може бути роздільний на чотири основні етапи.

На першому етапі, коли проектований об'єкт представляє складну інтегровану технічну систему, на основі принципу декомпозиції відбувається його розбиття на основні елементи, що становлять, і об'єкти. При цьому, наприклад при фізичному моделюванні, аналізуючи взаємозв'язок між окремими структурними елементами проектованого об'єкту сформулюються основні закономірності про зв'язки між ними в математичних термінах, які дозволяють здійснювати не тільки якісний, але і кількісний аналіз.

На другому етапі здійснюється рішення прямої або зворотної задачі проектування (тут доречно провести аналогію з першим і другим завданнями динаміки, завданнями аналізу і синтезу в теорії механізмів і машин). При рішенні прямої задачі по відомих зовнішніх і внутрішніх параметрах структурного елементу визначаються його вихідні параметри. Зворотне завдання проектування є складнішим. Тут по відомих зовнішніх і вихідних параметрах визначаються внутрішні параметри. Зворотне завдання відповідає в інженерній практиці проектувальному розрахунку, коли варіювання значень внутрішніх параметрів приводить до оптимізації деякої цільової функції (наприклад завдання синтезу механізмів, коли як цільова функція можна використовувати мінімальні габаритні розміри механізму або його масу). Зворотне завдання є складним ще і тому, що не завжди відома інформація про сформульовані якісні уявлення про внутрішні параметри проектованого об'єкту (наприклад процеси, що протікають в зоні формування тканин при взаємному переміщенні і деформації основних і утокових ниток; процеси протікають при транспортуванні пакету матеріалів на швейній машині; процеси тепло -

масо переносу при шліхтуванні основи в підготовчому виробництві і ін.). В даному випадку, коли зовнішні параметри (характеризуючи умови функціонування проектованого об'єкту) відомі і відомі відповідні їм вихідні параметри необхідно використовувати імітаційні моделі, побудовані за принципом чорного ящика. Особливу роль грають процедури синтезу при проектуванні складних технічних об'єктів. Після рішення прямої або зворотної задачі для кожного конкретного структурного елементу, з яких складається технічна система, необхідно здійснити інтегрований параметричний синтез, який полягатиме в стиковці зовнішніх і вихідних параметрів кожного з об'єктів, що становлять. Аналогія цьому процесу простежується наприклад при розгляді рівноваги системи, що складається з ***n*** матеріальних тіл. Для плоскої системи сил, для кожного з тіл тих, що входять в систему можна скласти 3 рівняння рівноваги. Таким чином, для визначення всіх внутрішніх реакцій в'язей необхідно буде скласти і спільно вирішити ***3n*** рівнянь рівноваги.

Третій етап включає аналіз отриманих результатів і порівняння їх з існуючими результатами, які виходять шляхом проведення експерименту на макеті або при натурному експерименті. Даний етап є дуже важливим, оскільки дозволяє встановити чи задовольняє прийнята гіпотетична модель критерію практики, чи порівняна в межах точності отримана інформація про об'єкт з результатами спостережень явищ, що вивчаються. Вельми істотним при цьому є вибір необхідної точності значення цільової функції, що набуває. Основним критерієм є абсолютна величина або діапазон зміни значень вихідних параметрів (наприклад, істотно різні вимоги до точності пред'являються при проектуванні окремих вузлів і механізмів у важкому і точному машинобудуванні).

Результати, отримані на третьому етапі, є основою для проведення четвертого, завершального етапу. У разі, коли результати аналізу математичної моделі співпадають в межах заданої точності з результатами

експериментальних спостережень, то можна вважати, що пропонована математична модель задовольняє вимогам, що пред'являються. Інакше виникає необхідність модернізації математичної моделі. Таким чином, тут доречно говорити про її ускладнення з погляду обліку додаткових зовнішніх і внутрішніх параметрів, виявленню додаткового взаємозв'язку між ними. Отримувана ускладнена математична модель, відповідно до принципів об'єктно-орієнтованого моделювання, успадковує властивості свого прототипу. Дана ієрархія математичних моделей (як структурна, так і функціональна) дозволяє істотно розширити і збагатити знання про проектований об'єкт.

Розробка математичних моделей технічних систем і технологічних процесів легкої і текстильної промисловості в першу чергу направлена на проведення процедури обчислювального експерименту. В зв'язку з цим є актуальним аналіз послідовності проведення окремих етапів процесу побудови математичної моделі і організації обчислювального експерименту. На рис.1.1 представлена спрощена блок-схема, яка визначає послідовність проведення проектувальних розрахунків технічних систем (ТС) і технологічних процесів легкої і текстильної промисловості на початковій стадії проектування. Дана спрощена блок-схема дозволяє враховувати як структурні характеристики проектованих технічних систем, так і властивості їх вхідних, вихідних і внутрішніх параметрів.

На початковому етапі проводиться аналіз проектованої технічної системи на предмет можливості її ділення на окремі елементарні технічні об'єкти. Професор Зарубін В.С. характеризує їх як об'єкти мікро рівня.

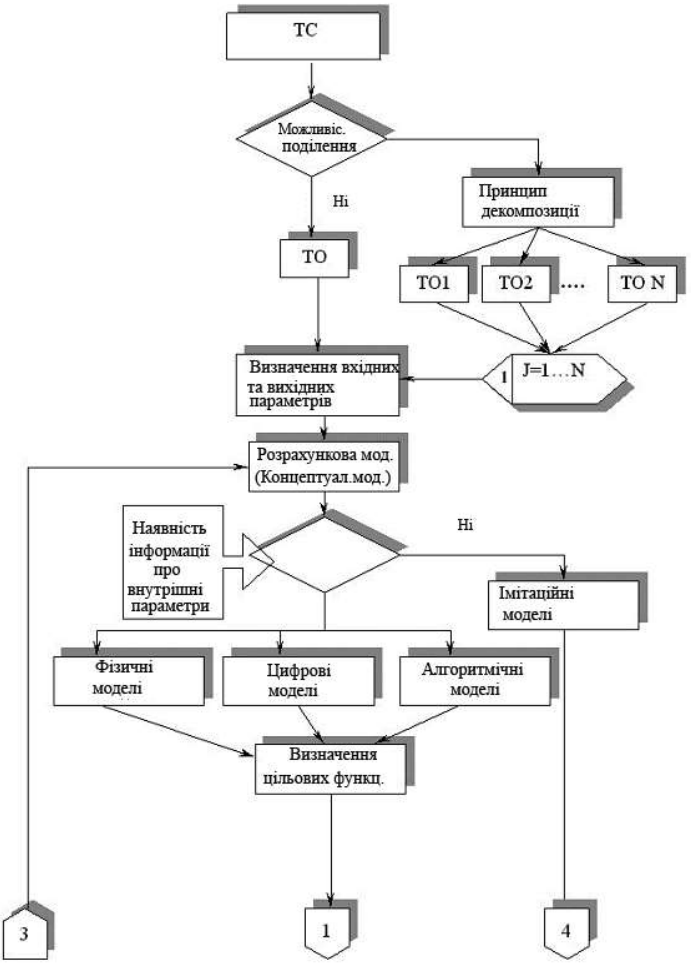
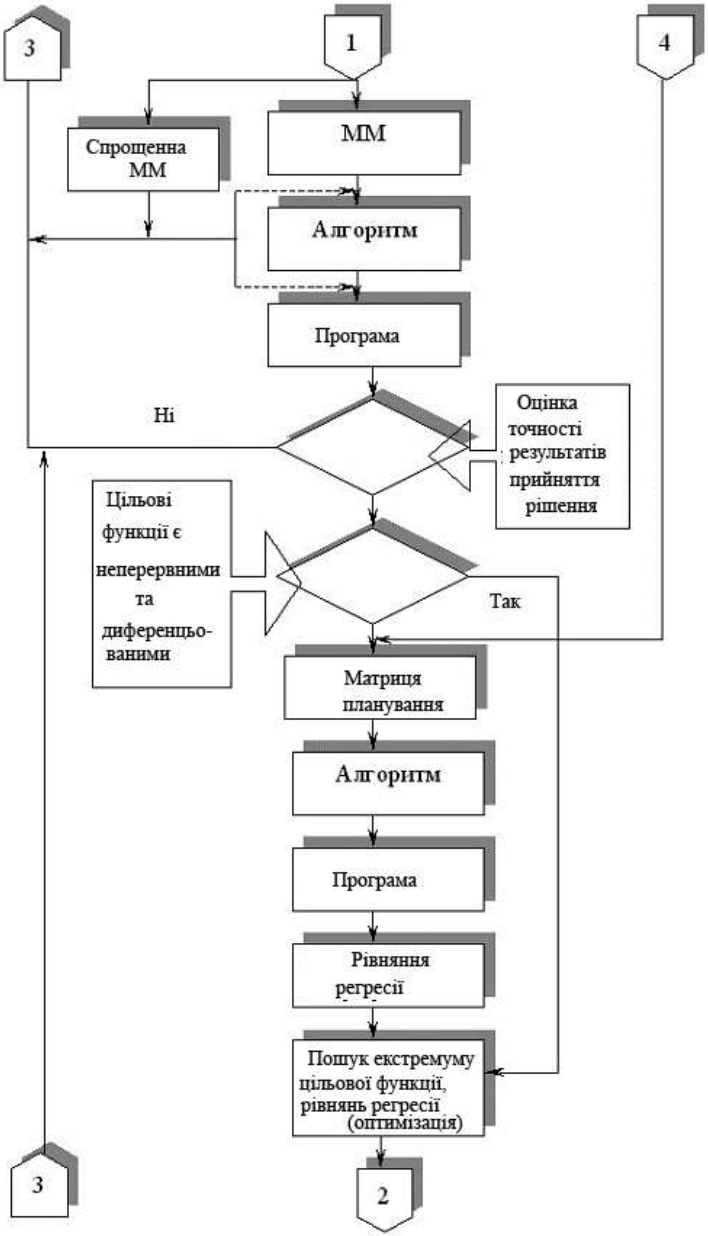
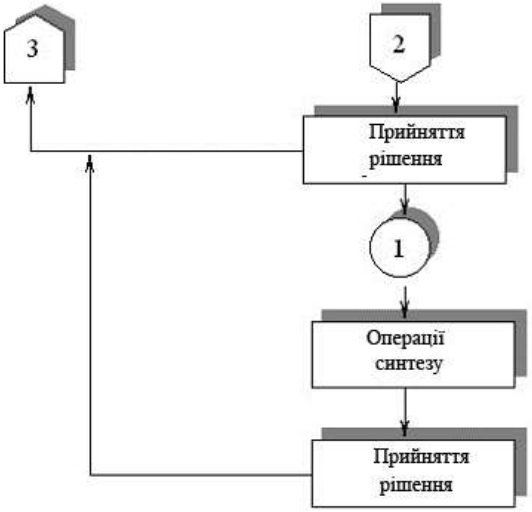


Рис.1.1. Блок-схема послідовності проведення проектувальних розрахунків



Продовження рис.1.1



Продовження рис.1.1

Детальніше питання структурної класифікації будуть розглянуті нижче, а на цьому етапі вважатимемо, що структурна ієрархія технічної системи може включати як об'єкти мікро-, так і об'єкти макрорівня.

У випадку якщо можливість ділення відсутня або недоцільна, то подальшому аналізу піддається технічний об'єкт, який буде еквівалентний технічній системі.

У випадку якщо можливість ділення відсутня або недоцільна, то подальшому аналізу піддається технічний об'єкт, який буде еквівалентний технічній системі. Інакше, використовуючи принцип декомпозиции, разбиваем техническую систему на ряд технических объектов $TO1, TO2, ...TON$, де N - число технічних об'єктів мікро рівня, з яких складається технічна система (наприклад, будь-який плоский механізм можна розбити на групи Асура, ступінь рухливості яких дорівнює нулю).

Наступним важливим етапом (див. рис.1.1) є визначення вхідних і вихідних (цільових функцій) параметрів, які впливають на технічний

об'єкт або є наслідком його реакції на дію вхідних і внутрішніх параметрів. Необхідно відзначити, що число вхідних і внутрішніх параметрів визначається ступенем їх впливу на вихідні параметри і необхідною точністю. Для технічної системи, що складається з ряду технічних об'єктів, для кожного з них може бути свій набір вхідних і внутрішніх параметрів, що пояснюється специфікою їх будови, фізичною природою процесів що протікають в них.

Визначившись з вибором вхідних і вихідних параметрів (для цього можна використовувати результати попередніх експериментів, експертні оцінки і ін.) можна приступати до побудови розрахункової схеми (концептуальній моделі) з використанням спеціальних прийомів і символів наочного графічного зображення (наприклад кінематичні і електричні схеми). Паралельно з цим визначається наявність інформації про внутрішні параметри технічного об'єкту. У разі, коли дана інформація відсутня, для подальшого аналізу необхідно використовувати імітаційні моделі, які побудовані за принципом чорного ящика. Для кожного вхідного в систему об'єкту в пам'яті ЕОМ підтримується таблиця його параметрів. Кожна елементарна взаємодія в системі реалізується у вигляді одного або декількох алгоритмів, що дозволяють розраховувати невідомі параметри взаємодії по заданих. Отриманий набір підпрограм доповнюється провідною програмою, що управляє процесом звернення до підпрограм відповідно до заданої структури системи.

За наявності інформації про внутрішні параметри можливе використання фізичних, цифрових і аналогових моделей. Фізичні моделі більш повно відтворюють властивості і поведінку досліджуваного технічного об'єкту. Дана модель широко використовується при дослідженні механізмів, машин і технологічних процесів легкої і текстильної промисловості. Це пояснюється тим, що науковою основою для вивчення властивостей і поведінки перерахованих вище об'єктів є

теоретична механіка, теорія механізмів і машин, механіка нитки, опір матеріалів, деталі машин, механічна технологія волокнистих матеріалів, які є по суті продовженням і розвитком окремих розділів курсу фізики.

Використання аналогових обчислювальних машин (АВМ) дозволило вивчати поведінку складних технічних систем на моделі, що має іншу фізичну природу чим у реального об'єкту або процесу. Тут можна відзначити широко відомі електромеханічні, електромагнітні аналогії, які дозволяють моделювати поведінку складних систем з використанням електричних ланцюгів. Аналогові моделі можуть бути як прямій, так і непрямій аналогії. По-перше поведінка різних по фізичній природі об'єктів описується подібними функціональними залежностями (наприклад другий закон Ньютона і закон Ома і ін.). У другому випадку, з використанням принципу декомпозиції, технічна система розбивається на об'єкти мікрорівня, поведінка яких моделюється певними функціональними залежностями. Вирішення отриманих трансцендентних, алгебри і диференціальних рівнянь здійснюється різними електронними вирішальними пристроями АВМ, які утворюють схеми аналогічні структурі даних рівнянь.

У разі, коли параметри стану технічної системи представляються у вигляді сукупності дискретних значень використовують цифрові моделі, які вирішуються на ЕОМ з використанням чисельних методів. Точність отримуваних результатів, в даному випадку, значно вище чим при використанні АВМ.

На наступному етапі, виходячи з проектного завдання, визначаються цільові функції, які можна розглядати як функціональні залежності загального вигляду між вхідними, внутрішніми і вихідними параметрами. Дана математична формалізація не дозволяє ще здійснювати якісний і кількісний аналіз. На цьому етапі відбувається вибір прямого або зворотного завдання проектування і визначаються параметри, які істотно

впливають на поведінку технічної системи. Даний етап закінчується побудовою математичної моделі з використанням отриманої раніше інформації і проектного завдання (див. рис.1.1).

Необхідно декілька слів сказати про основні властивості математичних моделей, які повинні задовольняти певним властивостям. Огляд літературних джерел дозволяє виділити наступні основні властивості математичних моделей: повнота, точність, адекватність, економічність, робастность, продуктивність, наочність. Зупинимося на аналізі даних показників.

Під повнотою математичної моделі мається на увазі ступінь її відповідності проектованому технічному об'єкту щодо вихідного параметра оптимізації, який визначається технічним завданням. Наприклад, в процесі проектування нас цікавить розробка такого технічного об'єкту, функціональні параметри якого істотно важливіші за його структурні параметри (отримання необхідної траєкторії руху робочого органу механізму, при цьому обмеження на масу, геометричні розміри і число ланок відсутні або грають другорядну роль і ін.).

Під точністю математичної моделі мається на увазі необхідність прийнятного збігу результатів визначення значення цільових функцій (вихідних параметрів), отриманих при виконанні обчислювального експерименту, із значеннями, визначеними за допомогою експерименту.

Адекватність математичної моделі полягає в її здатності відображати основні властивості технічного об'єкту у визначених технічним завданням межах зміни вхідних і внутрішніх параметрів.

Під економічністю математичної моделі маються на увазі витрати часу на виконання обчислювальних операцій і ефективність використання пам'яті ЕОМ. Даний показник безпосередньо пов'язаний з проблемою ефективності запропонованого алгоритму рішення задачі. Сучасні можливості обчислювальної техніки цілком задовольняють, з погляду

продуктивності, тому кругу завдань, які виникають при проектуванні машин і технологічних процесів легкої і текстильної промисловості.

Робасність визначає стійкість математичної моделі до різного роду погрішностям, які виникають при виконанні обчислювального експерименту. Ці погрішності виникають при діленні числа на число близьке до нуля, при відніманні близьких по модулю величин. Вирішення цієї проблеми також пов'язане з умінням дослідника якісно розробляти обчислювальний алгоритм.

Продуктивність математичної моделі залежить від якісно підготовленої початкової інформації, яка необхідна при виконанні обчислювального експерименту. Це дозволить ефективно проводити ті тільки якісний, але і кількісний аналіз явищ, що вивчаються, і процесів.

Останньою властивістю є наочність математичної моделі. Дана властивість дозволяє дослідникові на початковій стадії проектування оцінювати результати і своєчасно вносити корективи при проведенні обчислювального експерименту.

Подальші три блоки (див. рис.1.1) визначають собою попередній обчислювальний експеримент. Сюди входять блоки ММ (математична модель), алгоритм, програма. Окремо представлений блок – спрощена модель. Необхідність проведення попереднього обчислювального експерименту обумовлена вимогами, що пред'являються до точності обчислення дискретних значень цільової функції. Набутих значень порівнюються з відомими значеннями цільової функції, які можуть бути визначені в кінцевому числі точок області визначення цільового функціонала. У випадку якщо точність отримуваного результату (для вибраних крапок) задовольняє пред'явленим вимогам, то тоді математична модель може бути прийнята і використана для подальших обчислень. Інакше необхідно її удосконалити починаючи з розрахункової схеми. На початковому етапі можна зробити спробу аналізу технічного об'єкту з

використанням спрощеної математичної моделі, яка включає мінімальне число вхідних і внутрішніх параметрів. Процедура отримання кінцевого результату і його порівняння з відомими даними аналогічна випадку як і для математичних моделей вищого ієрархічного порядку. У разі низької точності також відбувається перехід до блоку, що відповідає за розробку розрахункової схеми, яку необхідно уточнити.

Блок, що включає алгоритм рішення, є відомими методами вирішення отриманих математичних залежностей для цільової функції (це можуть бути чисельні методи вирішення трансцендентних, диференціальних рівнянь, систем лінійних рівнянь і ін.). Для виконання відповідного алгоритму необхідна програма, яка дозволяє реалізувати його на ЕОМ.

Цілком очевидно, що якщо вихідні параметри пов'язані з вхідними і внутрішніми параметрами явним чином і цільовими функціоналами є безперервні функції, що диференціюються, на вибраних інтервалах варіювання вхідних і внутрішніх параметрів, то завдання пошуку екстремуму цільових функцій (оптимізація) не викликає великих утруднень. На блок-схемі (див. рис.1.1) це враховується оператором умовного порівняння.

Серйозніше завдання виникатиме тоді, коли вихідні параметри не можна безпосередньо виразити через вхідні і внутрішні параметри. Наприклад, якщо цільовий функціонал представлений трансцендентним рівнянням або диференціальним рівнянням, яке дуже складно про інтегрувати. В цьому випадку необхідно використовувати відповідні чисельні методи, які дозволяють визначити значення цільового функціонала для кінцевого числа дискретних значень варійованих змінних. В цьому випадку необхідно проводити основний обчислювальний експеримент з використанням методів активного планування експерименту. Тоді на основі проектного завдання складається відповідний план проведення експерименту, визначаються інтервали варіювання

вхідних і внутрішніх параметрів (змінних), визначаються значення цільових функцій для відповідних змінних з використанням розроблених алгоритмів і програмного забезпечення. Аналіз сказаного ще раз підкреслює важливість проведення попереднього експерименту, на стадії якого математична модель уточняється до рівня відповідного вимогам, що пред'являються до результатів.

На наступному етапі проводиться визначення коефіцієнтів в рівнянні регресії, визначаються значущі коефіцієнти, проводиться оцінка адекватності отриманого регресійного рівняння. Такий підхід дозволяє перейти від дискретних значень цільової функції до рівняння, яке з певною довірчою вірогідністю описує технічний об'єкт, що вивчається. При цьому отримувана функція, на заданому інтервалі варіювання змінних, буде безперервною і такою, що диференціюється. Після цього можна визначати екстремуми (глобальні або локальні) і проводити оптимізацію.

У випадку якщо отримувані результати не задовольняють проектному завданню, необхідно повернутися до коректування розрахункової схеми.

Для визначення значення цільової функції в характерних точках плану необхідно також розробляти алгоритм і програму або використовувати вже відомі, які були отримані при проведенні попереднього обчислювального експерименту.

При проведенні основного обчислювального експерименту виникають певні труднощі, які пов'язані з тим, що паралельні чисельні досліди, що виконуються в одній точці факторного простору, не дозволяють визначити математичне очікування і дисперсію значень цільового функціонала в характерних точках плану. За даними літературних джерел доцільно розглядати помилки дослідів як випадкові величини, створюючи множину, кількість елементів якого рівна кількості дослідів, і використовувати їх для оцінки адекватності рівняння регресії. Кожен елемент безлічі помилок визначається як різниця між обчисленими

на ЕОМ і отриманими по рівнянню регресії значенням цільового функціонала в досвіді.

Для технічних систем після блоку ухвалення рішення, з використанням операції циклу, відбувається перехід до технічного об'єкту ***To2*** і повторюється вся послідовність процедур описаних вище. Даний процес виконується до тих пір, поки не буде розглянутий останній технічний об'єкт ***TON*** . Потім слідують операції синтезу отриманих рішень по кожному технічному об'єкту і ухвалюється рішення щодо всієї технічної системи.

1.2. Класифікація математичних моделей

Класифікація всього різноманіття математичних моделей, які описують різні явища і об'єкти, є завданням вельми складної. Зважаючи на специфіку даної книги зупинимося на класифікації математичних моделей, які використовуються для дослідження технологічних процесів, а також для виконання аналізу і синтезу механізмів і машин легкої і текстильної промисловості. У основу даної класифікації можна покласти різні особливості і ознаки математичних моделей, на які указує проф. Зарубін В.С. Основними з них є ступінь деталізації технічного об'єкту, характер властивостей, що відображаються, і методи отримання і представлення математичної моделі.

На рис.1.2 представлена класифікація математичних моделей залежно від ступеня їх деталізації і з урахуванням характеру властивостей, що відображаються. Тут необхідно зробити декілька попередніх зауважень, які стосуються питань структурної ієрархії математичних моделей. Цілком очевидно, що елементарним технічним об'єктом може вважатися деяка система з розподіленими параметрами. Сукупність таких систем можна характеризувати як деяка множина, клас якої визначає його потужність.

Даний параметр може характеризуватися кардиналом (кардинальним числом) даної множини. Інтеграція даних систем дозволяє отримати новий об'єкт вищого класу. Наприклад, для технологічного устаткування елементарним технічним об'єктом може бути ланка (тверде тіло, що входить до складу механізму). Об'єктом вищого класу є механізм (система тіл, призначена для перетворення руху одного або декількох твердих тіл в необхідні рухи інших твердих тіл). Нарешті до вищого класу відноситься машина (пристрій, що виконує механічний рух для перетворення енергії, матеріалів і інформації з метою заміни або полегшення фізичної і розумової праці людини). На думку проф. Зарубіна В.С., для забезпечення найбільш загальної уніфікації, тут доречно говорити про моделі мікро-, макро- і мета рівня.

Як видно з рис.1.2 математичні моделі властивостей, що по характеру відображаються, можна класифікувати як моделі структурні і функціональні. Структурні моделі класифікуються на топологічні моделі $1...N$ (N - число топологічних моделей даного технічного об'єкту) і геометричні моделі $11..1N, N1...NN$. Топологічні моделі відносяться до вищого класу в структурній ієрархії і призначаються для з'ясування і уточнення взаємозв'язку між окремими елементами технічного об'єкту (представляються у вигляді графів, структурних схем, матриць, списків і ін.). Геометричні моделі відносяться до нижчого класу структурної ієрархії і містять відомості про форму і розміри елементів технічного об'єкту і про їх взаємне розташування.

Функціональні математичні моделі, як наголошувалося вище, можна розглядати як сукупність моделей мета- ($1...N$), макро- ($11...1N, N1...NN$) і мікро рівня ($111...11N, 1N1...1NN, N11...N1N, NN1...NNN$). При визначенні місця математичної моделі в даній класифікації, на попередньому етапі, визначається клас математичної моделі.

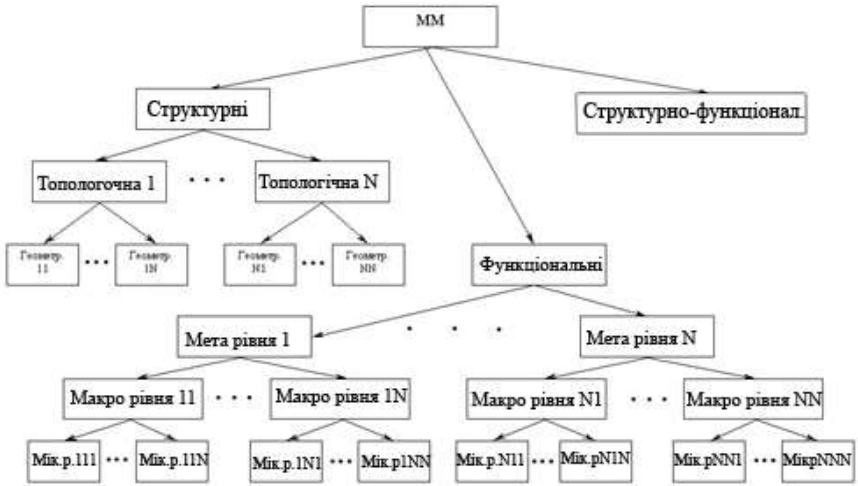


Рис.1.2. Класифікація математичних моделей залежно від ступеня їх деталізації

Потім визначається можливість додаткової деталізації на кожному з рівнів з відповідним кардинальним числом. Функціональні математичні моделі відображають фізичні, механічні, хімічні і ін. процеси, що відбуваються в технічних об'єктах.

Цілком очевидно, що для складних по своїй структурі об'єктів можливі інтегровані математичні моделі структурно-функционального характеру. Такі моделі можуть використовуватися для опису технічних об'єктів на макро- і мета рівнях.

На рис.1.3 представлена класифікація математичних моделей по способах отримання і уявлення. Тут можна виділити теоретичні,

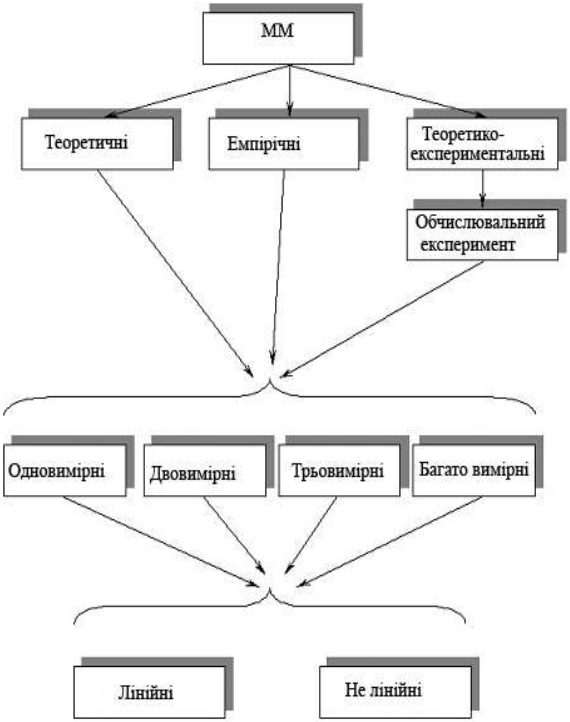


Рис.1.3. Класифікація математичних моделей по способах отримання і уявлення

емпіричні (досвідчені, експериментальні) і теоретико-экспериментальные моделі. Окремим випадком отримання теоретико-экспериментальной моделі можна вважати обчислювальний експеримент (про нього мовилося вище). Для кожного вказаного вище виду математичної моделі отримувана цільова функція може залежати від одного (лінійне завдання), два (плоске завдання), три (просторове завдання) і багатьох чинників (вхідних і внутрішніх параметрів). Крім того, дані залежності можуть носити лінійний і нелінійний характер.

Нарешті розглянемо класифікацію математичних моделей по характеру властивостей вхідних, вихідних і внутрішніх параметрів (див. рис.1.4).

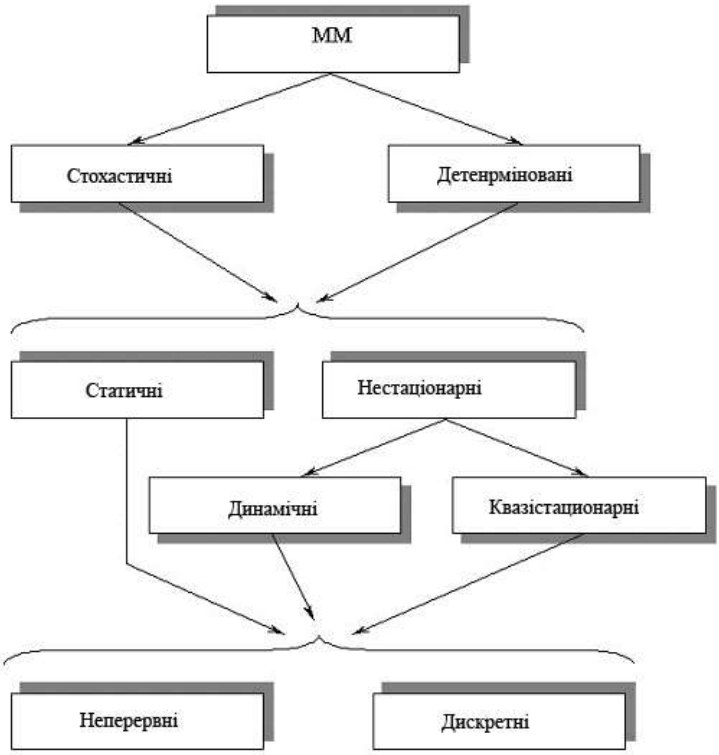


Рис.1.4. Класифікація математичних моделей по характеру властивостей вхідних, вихідних і внутрішніх параметрів

По характеру даних можна виділити стохастичні (серед даних, що представляються, містяться випадкові величини) і детерміновані (визначені) математичні моделі. Можливий варіант, коли містяться дані двох типів, тоді говорять про змішані моделі. Змішані моделі можуть відноситися тільки до моделей макро- і метарівня.

По характеру зміни параметрів від часу розрізняють статичні і нестаціонарні моделі. Останні підрозділяються на динамічних і

квазістатичних. У разі статичних моделей параметри не залежать від часу. Коли враховуються інерційні властивості технічних об'єктів необхідно використовувати динамічні моделі. Квазістатичні моделі відповідають випадку, коли параметри змінюються від часу з дуже малою швидкістю (наприклад, зміна діаметру навою основов'язальних і текстильних машин).

Значень цільових функціоналів моделі, що по характеру набувають, підрозділяються на моделі, де дані носять безперервний характер і на моделі з дискретними значеннями вхідних параметрів. На макро- і мета рівні можуть бути випадки, коли моделі носитимуть змішаний характер.

1.3. Математичні моделі, що реалізовуються з використанням чисельних методів

1.3.1. Системи лінійних рівнянь

При проведенні статичних і динамічних досліджень механічних систем доводиться стикатися з необхідністю вирішення систем лінійних рівнянь. Тут можна говорити про їх використання не тільки при вивченні рівноваги механічних об'єктів, але і при вирішенні систем диференціальних рівнянь як звичайних, так і в приватних похідних механічних систем, що описують поведінку, з великим числом мір свободи.

У загальному випадку систему лінійних рівнянь можна представити таким чином

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k + \ldots + a_{1n}x_n = b_1, \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k + \ldots + a_{2n}x_n = b_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mk}x_k + \ldots + a_{mn}x_n = b_m, \end{aligned} \qquad (1.3.1.1)$$

де a_{11} … a_{nn} – коефіцієнти при невідомих величинах; x_1 … x_n – невідомі (при вирішенні завдань статики і динаміки це можуть бути реакції зв'язків, узагальнені координати, швидкості і прискорення); b_1 … b_m – постійні

величини(необхідно відзначити, що враховуючи фізичну суть досліджуваних статичних і динамічних процесів розмірність парних творів в лівій частині рівності системи (1.3.1.1) повинна співпадати з розмірністю постійних $b_1 \dots b_m$); n – число невідомих, які входять в систему; m – число рівнянь.

Систему рівнянь (1.3.1.1) зручно представити у векторній формі

$$A\bar{x} = \bar{b}, \qquad (1.3.1.2)$$

де A – матриця коефіцієнтів m х n; $\bar{x}$ - шуканий n – компонентний вектор; $\bar{b}$ - заданий m – компонентний вектор.

Враховуючи, що нижче розглядатимуться тільки статично визначні завдання (число невідомих дорівнює числу можливих рівнянь $m=n$), то систему лінійних рівнянь (1.3.1.1) можна представити у вигляді (матриця коефіцієнтів перетвориться до квадратного вигляду)

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \dots + a_{1k}x_k + \dots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2k}x_k + \dots + a_{2n}x_n = b_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nk}x_k + \dots + a_{nn}x_n = b_n. \end{gathered} \qquad (1.3.1.3)$$

Визначник (детермінант) матриці коефіцієнтів A визначимо рівністю

$$\Delta = det\, A = \begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{Vmatrix} \qquad (1.3.1.4)$$

Тут необхідно відзначити, що у разі, коли $\Delta \neq 0$, то матриця називається не виродженою. У разі, коли заданий компонентний вектор $\bar{b}=0$ – система (1.3.1.3) називається однорідною і має ненульове рішення при $\Delta = 0$. Якщо система рівності (1.3.1.3) має рішення, то вона називається сумісною. Інакше вона називатиметься суперечливою або несумісною. За наявності єдиного вирішення системи (1.3.1.3) можна говорити про те, що система є

визначеною. За наявності двох і більш за рішення система буде невизначеною.

Випадок, коли детермінант $\Delta \neq 0$ забезпечує існування єдиного рішення.

Перейдемо до визначення методів вирішення системи рівнянь (1.3.1.3). Точне рішення в явному виді даної системи може бути отримане з використанням формули Крамера. Суть методу полягає в послідовному діленні перетвореного визначника (у якого відповідний стовпець коефіцієнтів лівої частини системи замінюється стовпцем ***b1.... bn***) на початкового визначника, що складається з коефіцієнтів лівої частини системи рівнянь. Для визначення невідомих ***x1 ..... xn*** необхідно виконати ***n*** перестановок

$$x_1 = \frac{\begin{Vmatrix} b_1 & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ b_2 & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ b_n & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{Vmatrix}}{\begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{Vmatrix}},$$

$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$

$$x_k = \frac{\begin{Vmatrix} a_{11} & \dots & a_{1k-1} & b_1 & a_{1k+1} & \dots & a_{1n} \\ a_{21} & \dots & a_{2k-1} & b_2 & a_{2k+1} & \dots & a_{2n} \\ a_{n1} & \dots & a_{nk-1} & b_n & a_{nk+1} & \dots & a_{nn} \end{Vmatrix}}{\begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{Vmatrix}}, \qquad (1.3.1.5)$$

$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$

$$x_n = \frac{\begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & b_1 \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & b_2 \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & b_n \end{Vmatrix}}{\begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{Vmatrix}}.$$

Даний метод з обчислювальної точки зору є не ефективним. Для визначення всього коріння ***x1 ... xn*** необхідно виконати з використанням

системи (1.3.1.5) $(n^2 - 1)\cdot(n!)$ множень і n ділень. Для системи складається з 50 рівнянь необхідно буде виконати $7{,}610^{67}$ операцій множення. Виконання такого числа операцій в рамках реальної тимчасової протяжності недоступно сучасним ЕОМ. Як відомо, рівновага одного тіла під дією просторової системи сил описується 6 рівняннями. Для дослідження рівноваги механічної системи з 9 тіл буде потрібно 54 рівняння рівноваги з 54 невідомими. А сучасні машини легкої і текстильної промисловості і їх окремі вузли складаються із значно більшого числа ланок.

Виходячи з цього ефективними методами визначення невідомих в системі рівнянь (1.3.1.3) можна вважати прямі і ітераційні методи. Використання перших дозволяє за кінцеве число дій отримати точне вирішення системи, якщо всі величини задані і обчислення проводяться точно, без помилок при округленні результату. Ітераційні методи полягають в отриманні наближеного вирішення системи лінійних рівнянь із заданою точністю шляхом побудови ітераційної послідовності, що сходиться до шуканого рішення.

Вибір того або іншого методу залежить від характеристик ЕОМ, числа рівнянь системи, характеристик матриці коефіцієнтів A. Наприклад, при складанні рівнянь рівноваги твердого тіла деякі коефіцієнти aij дорівнюють нулю. У таких випадках використовувати метод Гауса послідовного виключення невідомих і метод Гауса з вибором головного елементу не можна. Тому, що при реалізації алгоритму на одному з етапів обчислень відбувається ділення попереднього результату на 0. Машина видає повідомлення про виконання некоректної операції.

Нижче ми розглянемо декілька різних методів на конкретних прикладах і постараємося показати як сильні, так і слабкі сторони кожного з них.

Метод послідовного виключення невідомих Гауса заснований на послідовному пониженні порядку системи (1.3.1.3) за рахунок виключення невідомих ***x1 ... xn-1*** в лінійних рівняннях. Це приводить до отримання матриці коефіцієнтів ***aij*** трикутного вигляду. Ця послідовність процедур характеризується як прямий хід виключення змінних. Для його здійснення необхідно послідовно віднімати перше рівняння з подальших, умножаючи його ліву і праву частини на постійні коефіцієнти, що представляють приватне від ділення постійних коефіцієнтів $m_{ji}=a_{ji}/a_{ii}$, де $i = 1,2,\ldots n-1; j= i + 1, i +2,\ldots n$. Кінцева система рівнянь на прямому ході має вигляд

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k + \ldots + a_{1n}x_n = b_1, \\ &a_{22}^{(1)}x_2 + \ldots + a_{2k}^{(1)}x_k + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{kk}^{(k-1)}x_k + \ldots + a_{kn}^{(k-1)}x_n = b_k^{(k-1)}, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{nn}^{(n-1)}x_n = b_n^{(n-1)}, \end{aligned} \qquad (1.3.1.6)$$

де

$$a_{22}^{(1)} = a_{22} - \frac{a_{21}}{a_{11}}a_{12}; \ldots a_{2k}^{(1)} = a_{2k} - \frac{a_{21}}{a_{11}}a_{1k}; \ldots a_{2n}^{(1)} = a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}; b_2^{(1)} = b_2 - \frac{a_{21}}{a_{11}}b_1;$$

$$a_{kk}^{(k-1)} = a_{kk}^{(k-2)} - \frac{a_{kk-1}^{(k-2)}}{a_{k-1k-1}^{(k-2)}}a_{k-1k-1}^{(k-2)}; \ldots a_{kn}^{(k-1)} = a_{kn}^{(k-2)} - \frac{a_{kk-1}^{(k-2)}}{a_{k-1k-1}^{(k-2)}}a_{k-1n}^{(k-2)}; b_k^{(k-1)} = b_k^{(k-2)} - \frac{a_{kk-1}^{(k-2)}}{a_{k-1k-1}^{(k-2)}}b_{k-1}^{(k-2)};$$

$$a_{nn}^{(n-1)} = a_{nn}^{(n-2)} - \frac{a_{nn-1}^{(n-2)}}{a_{n-1n-1}^{(n-2)}}a_{n-1n}^{(n-2)}; b_n^{(n-1)} = b_n^{(n-2)} - \frac{a_{nn-1}^{(n-2)}}{a_{n-1n-1}^{(n-2)}}b_{n-1}^{(n-2)}.$$

Блок-схема програми реалізації методу Гауса послідовного виключення невідомих показана на рис.1.3.1.1.

Декілька слів необхідно сказати про недоліки даного методу. Розглянемо систему лінійних рівнянь

$$\begin{aligned} &1\cdot 10^{-11}x_1 - 3x_2 + 2x_3 = 1, \\ &1\cdot 10^{10}x_1 - 5x_2 + 4x_3 = 2, \\ &10x_1 - 7x_2 + 3x_3 = 9. \end{aligned} \qquad (1.3.1.7)$$

Вирішення (1.3.1.7) з використанням методу Гауса послідовного виключення невідомих дає (обмежуємося чотирма значущими цифрами після коми)

$$x_1 = 0{,}0000 \;; x_2 = -2{,}6667; x_3 = -3{,}5000 \;.$$

Переставимо рівняння системи (1.3.1.7) місцями. Друге на місце першого, третє на місце другого, а перше на місце третього. Знову визначимо коріння

$$x_1 = 0{,}0000 \;; x_2 = -3{,}0000; x_3 = -4{,}0000 \;.$$

Як видно, останній результат дозволяє отримати тотожності для першого і третього рівнянь системи (1.3.1.7). Отримане раніше коріння не дозволяє цього. Отже, можна говорити про те, що порядок рівнянь може впливати на кінцевий результат. Якби можна було здійснити ідеальний обчислювальний процес без помилок округлення (що присутній в реальному обчислювальному процесі), то незалежно від порядку рівнянь в обох випадках отримали б одну і ту ж відповідь.

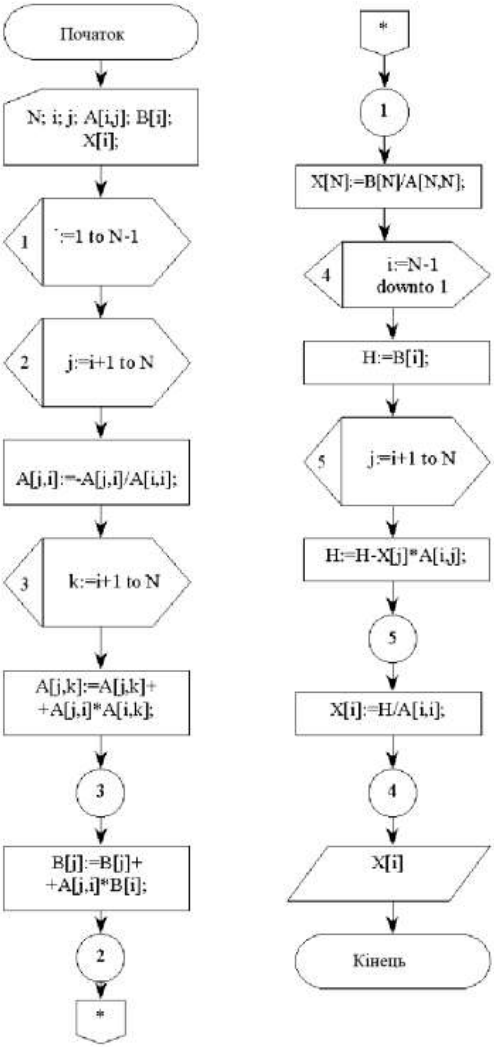
Якщо ми при обчисленнях обмежимося не 4, а 11 значущими цифрами після коми, то тоді останній результат матиме вигляд

$$x_1 = 0{,}00000000030 \;; x_2 = -2{,}99999999880; x_3 = -3{,}99999999820 \;.$$

При такому результаті після підстановки в (1.3.1.7) всі три рівняння звертаються в тотожність.

Така втрата точності пояснюється дуже маленьким значенням коефіцієнта a_{11} , що приводить до різкого зростання коефіцієнтів $a^{(1)}{}_{22} \ldots a^{(1)}{}_{2n}$; $b^{(1)}{}_2$; $\ldots a^{(1)}{}_{n2} \ldots a^{(1)}{}_{nn}$; $b^{(1)}{}_n$ на першому кроці прямого ходу. Для виключення цього використовуються різні схеми перетворення матриць коефіцієнтів на початковому етапі перед визначенням коріння. На рис.1.3.1.2 і 1.3.1.3 представлені блок-схеми програм, що реалізовують два методи подібного перетворення. Перший з них полягає в перетворенні матриці коефіцієнтів

на основі вибору головного (ведучого) елементу, який по модулю має максимальне значення серед інших.



Блок-схема PROGRAM S_G_PE

На першому кроці вибирається максимальний по модулю елемент a_{ij} . Даний рядок матриці переставляється на перше місце $a_{1j} = a_{ij}$. Потім

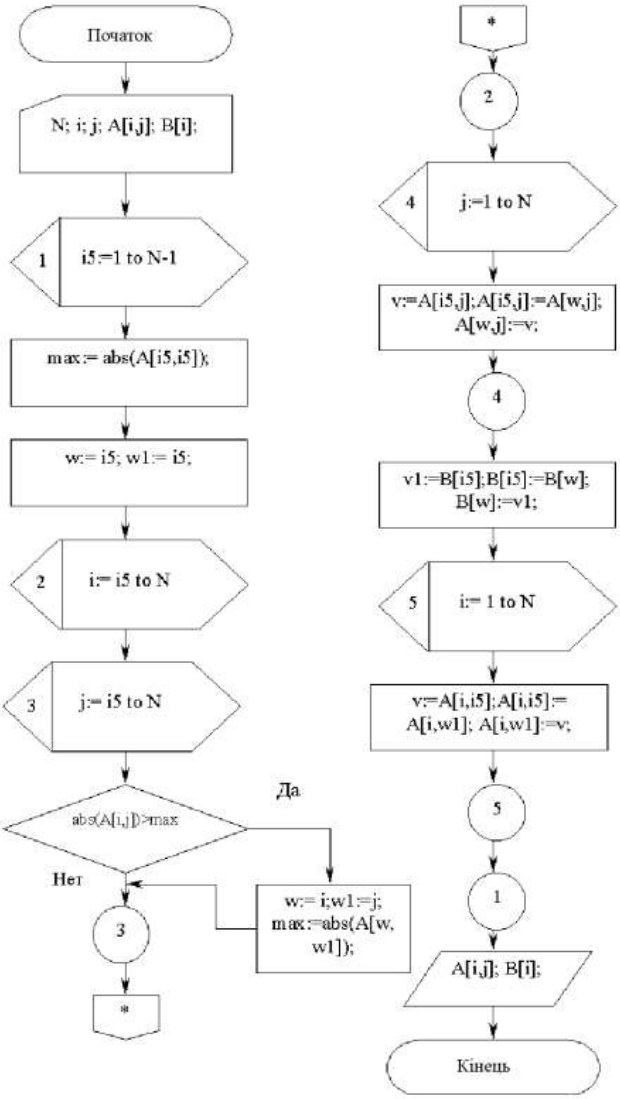
стовпець, що містить максимальний по модулю елемент $max = abs(a_{ij})$, міняється місцями з першим стовпцем $a_{11} = a_{ij}$. Таким чином, елемент a_{ij} переміщається в крайню ліву позицію на головній діагоналі матриці коефіцієнтів. На другому кроці розглядають ту, що залишилася $n-1$ систему рівнянь. Знову визначається максимальний по модулю елемент і переміщається на другу позицію на головній діагоналі. Метод вирішення системи лінійних рівнянь, побудований на такому перетворенні матриці коефіцієнтів, носить назву методу Гауса з вибором головного елементу.

Інший метод полягає в наступному (див. рис.1.3.1.3): на першому кроці підсумовуються абсолютні значення всіх коефіцієнтів матриці по рядках $\sum_{j=1}^{n} abs(a_{ij})$. Після цього відбувається їх порівняння і рядки матриці розташовуються в порядку убування суми $s1$. На другому кроці відбувається підсумовування модулів всіх елементів матриці коефіцієнтів по стовпцях. Після цього відбувається порівняння отриманих сум, і стовпці розташовуються зліва на право в порядку убування сум.

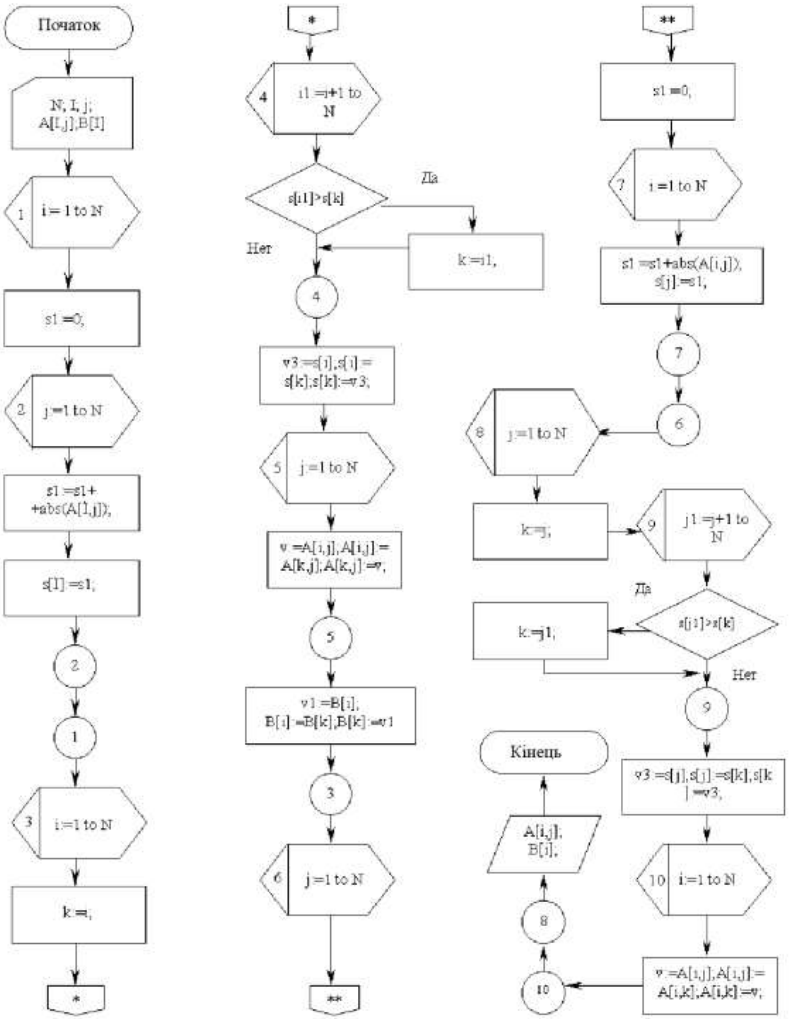
З великими складнощами доводиться стикатися, коли ряд коефіцієнтів матриці дорівнює нулю. В цьому випадку може виникнути ситуація, коли машина виконає не коректну операцію (ділення на нуль) і виконання програми буде припинено. Для виключення цього можна використовувати метод обертання, який дозволяє приводити систему лінійних рівнянь (1.3.1.3) до системи лінійних рівнянь з правою трикутною матрицею на прямому ходу по наступних формулах

$$M_{ki}\left(\sum_{j=1}^{n} a_{ij}x_j - b_i\right) - L_{ki}\left(\sum_{j=1}^{n} a_{kj}x_j - b_k\right) = 0,$$
$$L_{ki}\left(\sum_{j=1}^{n} a_{ij}x_j - b_i\right) - M_{ki}\left(\sum_{j=1}^{n} a_{kj}x_j - b_k\right) = 0, \qquad (1.3.1.8)$$
$$M_{ki} = \frac{a_{ii}}{\sqrt{a_{ii}^2 + a_{ki}^2}}, \quad L_{ki} = -\frac{a_{ki}}{\sqrt{a_{ii}^2 + a_{ki}^2}}.$$

де $i=1,2,\ldots n-1;\ k=i+1,\ldots n.$



Блок-схема PROGRAM PRE_MATR_VGE



Блок-схема PROGRAM PRE_MAT_MAX_SUM

У випадку якщо $a_{ii} = 0$, $a_{ki} \neq 0$, то $M_{ki} = 0$, $L_{ki} = 1$. Коли $a_{ii} = a_{ki} = 0$, то $M_{ki} = 1$ і $L_{ki} = 0$.

Ітераційні методи вирішення системи лінійних рівнянь розглянемо на прикладі методу простих ітерацій і методу Зейделя.

Розглянемо метод простих ітерацій, реалізація якого здійснюється з використанням наступної формули ітераційного процесу

$$x_{i(j+1)} = x_{i(j)} - \frac{1}{a_{ii}}\left(\sum_{k=1}^{n} a_{ik} x_{k(j)} - b_i\right). \qquad (1.3.1.9)$$

Обчислення продовжуються до тих пір, поки $xi_{(j+1)}$-$xi(j)>\varepsilon$, де - задана погрішність обчислень коріння системи лінійних рівнянь.

Декілька слів необхідно сказати про достатню умову збіжності методу простих ітерацій. Якщо

$$C = \max_{1 \le i \le n}(C_i) < 1, \qquad (1.3.1.10)$$

де $C_i = \left(\sum_{j=1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| - 1\right)$, $i=1,2,\ldots n$, то система лінійних рівнянь (1.3.1.3) має єдине рішення і ітераційний процес сходиться до рішення із швидкістю геометричної прогресії. Умову (1.3.1.10) можна інтерпретувати і так, ітераційний процес сходиться, якщо величина модуля кожного діагонального елементу матриці коефіцієнтів більше суми модулів решти елементів. Така умова вимагає, щоб матрицю коефіцієнтів (перед визначенні коріння системи лінійних рівнянь) перетворити до вигляду, коли на головній діагоналі матриці розташовуватимуться максимальні по модулю елементи.

Метод Зейделя організовує ітераційний процес по наступній формулі

$$x_{i(j+1)} = x_{i(j)} - \frac{1}{a_{ii}}\left(\sum_{k=1}^{i-1} a_{ik} x_{k(j+1)} + \sum_{k=i}^{n} a_{ik} x_{k(j)} - b_i\right). \qquad (1.3.1.11)$$

Його відмінність від методу простих ітерацій полягає в тому, що уточнені значення $xi_{(j+1)}$ відразу підставляються в подальші рівняння.

Достатньою умовою збіжності методу Зейделя є наступний вираз

$$\sum_{\substack{j=1 \\ j \ne i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \le q, \quad i = 1...n, \quad q < 1. \qquad (1.3.1.12)$$

Зазвичай метод Зейделя сходиться швидше, ніж метод простих ітерацій.

Декілька слів необхідно сказати про обумовленість матриці коефіцієнтів, яка визначається з виразу

$$\mu = \frac{sup\left[\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n} a_{ij}x_j\right)^2}\right]}{inf\left[\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n} a_{ij}x_j\right)^2}\right]} \geq 1. \qquad (1.3.1.13)$$

Якщо обумовленість матриці коефіцієнтів μ близька до 1, то для такої системи відносна помилка при відшуканні коріння рівнянь порівнянна з відносною погрішністю, з якою задається права частина. Система буде погано обумовленою якщо $>\mu>1$.

Розглянемо простій приклад. Хай дана система двох рівнянь

$$2x_1 + 0x_2 = 2,$$
$$5x_1 + 0{,}05x_2 = 5.$$

Вирішення даної системи рівнянь має вид $x1=1,\ x2=0.$ Трохи змінимо праву частину другого рівняння, тоді

$$2x_1 + 0x_2 = 2,$$
$$5x_1 + 0{,}05x_2 = 5{,}05.$$

Вирішення нової системи рівнянь матиме вид $x1=1,\ x2=1.$ Незначна зміна правій частині другого рівняння привела до істотної зміни рішення. Визначимо по (1.3.1.13) обумовленість матриці коефіцієнтів останньої системи

$$\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n} a_{ij}x_j\right)^2} = \sqrt{29x_1^2 + 0{,}5x_1x_2 + 2{,}5\cdot 10^{-3}x_2^2}. \qquad (1.3.1.14)$$

Знайдемо найбільше і найменше значення функції (1.3.1.14) на одиничному колі із заміною $x1=\cos\alpha,\ x2=\sin\alpha.$ Тоді

$$\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n} a_{ij}x_j\right)^2} = \sqrt{29\cos^2\alpha + 0{,}5\cos\alpha\sin\alpha + 2{,}5\cdot 10^{-3}\sin^2\alpha}. \qquad (1.3.1.15)$$

Пошук безумовного екстремуму функції (1.3.1.15) однієї змінної дозволив визначити

$$\sup\left[\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}a_{ij}x_j\right)^2}\right]=5{,}385,$$
$$\inf\left[\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}a_{ij}x_j\right)^2}\right]=0{,}0186\ . \qquad (1.3.1.16)$$

Підставляємо результати (1.3.1.16) в (1.3.1.13) і визначаємо обумовленість матриці коефіцієнтів

$$\mu=\frac{5{,}385}{0{,}0186}=289{,}54\ .$$

Отриманий результат дозволяє зробити вивід про те, що погана обумовленість матриці коефіцієнтів приводить до істотно різних вирішень розглянутої системи двох лінійних рівнянь при незначній зміні правій частині другого рівняння на ***0,05.***

1.3.2. Трансцендентні і алгебраїчні рівняння

Важливу роль в історії математики, в розвитку її ідей і методів зіграли рівняння. Вони знаходять найширше розповсюдження при вирішенні теоретичних і прикладних завдань САПР устаткування і технологічних процесів легкої і текстильної промисловості.

Методи вирішення лінійних рівнянь були відомі ще стародавнім грекам. Коріння квадратного рівняння (рівняння алгебри 2-го ступеня)

$$ax^2+bx+c=0, \qquad (1.3.2.1)$$

де ***a, b, c*** – коефіцієнти рівняння (1.3.2.1); ***x*** – невідоме; визначаються по формулах

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}\ ,\ x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}. \qquad (1.3.2.2)$$

Залежно від знаку дискримінанта $D=b^2-4ac$ маємо: два дійсних і різних кореня ($D\rangle 0$); два комплексні корені ($D\langle 0$); один кратний корінь ($D=0$) $-b/2a$. Формули Вієта

$$x_1+x_2=-\frac{b}{a},\ x_1x_2=\frac{c}{a}, \qquad (1.3.2.3)$$

зв'язують між собою коріння і коефіцієнти квадратного рівняння (1.3.2.1). Теорема Вієта, для позитивного коріння, встановлює справедливість співвідношень між ними і коефіцієнтами многочлена

$$P(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0, \qquad (4.1.3.2)$$

у вигляді

$$\begin{gathered} a_0 = (-1)^n x_1 x_2 \ldots x_n, \\ a_1 = (-1)^{n-1}(x_1 x_2 \ldots x_{n-1} + x_1 \ldots x_{n-2} x_n + \ldots + x_2 x_3 \ldots x_n), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n-2} = x_1 x_2 + x_1 x_3 + \ldots + x_{n-1} x_n, \\ a_{n-1} = -(x_1 + x_2 + \ldots + x_n). \end{gathered} \qquad (1.3.2.5)$$

Перші вирішення квадратних рівнянь (1.3.2.1) з позитивними коефіцієнтами (для позитивного коріння) виявлені ще в працях Мухаммеда аль Хорезмі і Діофанту. У *XVI* столітті М.Штіфель запропонував узагальнений метод рішення (1.3.2.1) включаючи і негативне коріння. Остаточне рішення, з урахуванням випадку уявного коріння, запропоноване Р.Бомбеллі і С.Стевіном.

У рівнянні (4.1.3.2) n називають ступенем рівняння. Вирішення рівняння $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = 0$ полягає в знаходженні всього його коріння, лежачого в даній області значень невідомого. Вирішення рівнянь третього (n = 3) і четвертого (n = 4) ступенів було отримане італійськими математиками Дж. Кардано, Н.Тарталья і Л.Феррарі в *XVI* столітті.

Рівняння алгебри 3-го ступеня

$$ax^3 + bx^2 + cx + d = 0, \quad a \neq 0, \qquad (1.3.2.6)$$

приводимо до канонічного вигляду

$$y^3 + py + q = 0, \qquad (1.3.2.7)$$

де $y = x + \frac{b}{3a}$; $p = -\frac{b^2}{3a^2} + \frac{c}{a}$; $q = \frac{2b^3}{27a^3} - \frac{bc}{3a^2} + \frac{d}{a}$.

Коріння рівняння (1.3.2.7) визначається по формулах Дж. Кардано

$$y_1 = A + B,\quad y_{2,3} = -\frac{A+B}{2} \pm i\frac{A-B}{2}\sqrt{3},$$
$$A = \sqrt[3]{-\frac{q}{2} + \sqrt{Q}},\quad B = \sqrt[3]{-\frac{q}{2} - \sqrt{Q}}, \qquad (1.3.2.8)$$
$$Q = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2.$$

У якості A і $У$ вибираються будь-які значення кубічного коріння, що задовольняє рівності $AB = -p/3$.

Метод Л. Феррарі вирішення рівнянь 4-го ступеня зводиться до рішення одного кубічного і двох квадратних рівнянь. Рівняння четвертого ступеня

$$ax^4 + bx^3 + cx^2 + dx + e = 0, \qquad (1.3.2.9)$$

шляхом заміни $y = x + \frac{b}{4a}$ приводиться до неповного вигляду

$$y^4 + py^2 + qy + r = 0. \qquad (1.3.2.10)$$

Коріння рівняння (1.3.2.10) визначається по формулах

$$y_1 = \frac{1}{2}\left(\sqrt{z_1} + \sqrt{z_2} + \sqrt{z_3}\right),\quad y_2 = \frac{1}{2}\left(\sqrt{z_1} - \sqrt{z_2} - \sqrt{z_3}\right),$$
$$y_3 = \frac{1}{2}\left(-\sqrt{z_1} + \sqrt{z_2} - \sqrt{z_3}\right),\quad y_4 = \frac{1}{2}\left(-\sqrt{z_1} - \sqrt{z_2} + \sqrt{z_3}\right). \qquad (1.3.2.11)$$

де z_1, z_2, z_3 - коріння кубічної резольвенти

$$z^3 + 2pz^2 + (p^2 - 4r)z - q^2 = 0\ .$$

Знаки перед корінням в системі (1.3.2.11) вибираються так, щоб виконувалася умова $\sqrt{z_1}\sqrt{z_2}\sqrt{z_3} = -q$.

Безрезультатні спроби (що тривали близько 300 років) завершилися в *XIX* в., направлені на вирішення рівнянь 5-ої і вищих ступенів, дозволили норвезькому математикові Н.Абелю сформулювати відому теорему: ні для якого ***n,*** більшого або рівнішого п'яти, не можна вказати формулу, яка виражала б коріння будь-якого рівняння ***n-й*** ступеня через його коефіцієнти за допомогою радикалів і арифметичних дій.

Ще більші складнощі виникають при вирішенні рівнянь не алгебри (трансцендентних), у яких визначити явні вирази для коріння не представляється можливим.

Вирішення трансцендентних рівнянь $f(x)=0$ рекомендується здійснювати в два етапи. На першому етапі проводиться приблизне визначення коріння. Звичайно це можна здійснити графічним способом, а на другому етапі проводиться уточнення значення коріння. При цьому вважаємо, що трансцендентне рівняння або рівняння на деякому відрізку $[a,b]$ відповідають теоремі про існування кореня безперервної функції: якщо функція $f(x)$ безперервна на відрізку $[a,b]$, її похідна $f'(x)$ в $]a,b[$ зберігає знак, на кінцях відрізання функція приймає значення з протилежними знаками $f(a)f(b)\langle 0$, то на цьому відрізку існує корінь рівняння $f(x)=0$ (принаймні, хоч би один).

На рис.1.3.2.1а представлені графічні залежності двох функцій

$$y = ax,$$
$$y = e^{\arccos\left(\frac{1}{x}\right)},$$

трансцендентного рівняння

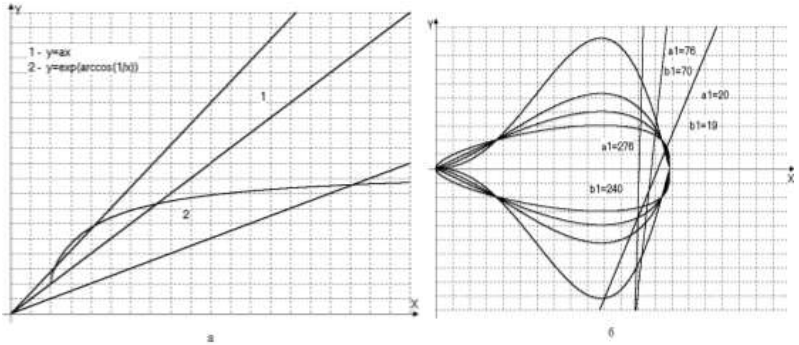
$$ax = e^{\arccos\left(\frac{1}{x}\right)}. \qquad (1.3.2.12)$$

Таку форму рівнянь отримуємо при вирішенні деяких завдань теорії механізмів і машин, механіки нитки. Рівняння прямих побудовані для різних значень коефіцієнта $a\left(0\langle a\langle 2\right)$. Точки перетину і відповідатимуть вирішенню рівняння (1.3.2.12). Причому, на вибраному інтервалі зміни x, для різних значень коефіцієнтів a, може бути від 1 до 2 коріння.

При визначенні величини кута прибою уточної нитки на ткацьких верстатах приходимо до системи двох трансцендентних рівнянь

$$\begin{aligned} bx^2 - x^4 - y^a &= 0, \\ a_1 x - b_1 - y &= 0. \end{aligned} \qquad (1.3.2.13)$$

На рис.1.3.2.1б представлено графічне (наближене) визначення коріння системи (1.3.2.13) для різних значень a, b, a_1, b_1. Їх величина залежить від ряду технологічних параметрів і конструктивних особливостей ткацького верстата.



Рiс.1.3.2.1. Графічне визначення коріння трансцендентного рівняння $f(x)=0$

Якщо в процесі досліджень приходимо до систем, що містять більше двох трансцендентних рівнянь, то графічне дослідження стає скрутним, а іноді і неможливим.

Необхідно відзначити, що вимога безперервності функції $f(x)$ на відрізку $[a,b]$ істотно. Непорушною вимогою є відсутність точок розриву.

Сформульована вище теорема існування може доводитися двома способами: з використанням методу фактичної побудови рішення; неконструктивний, ґрунтуючись на міркуваннях від осоружного - ланцюг логічних висновків показує, що вирішення трансцендентного рівняння повинне існувати інакше виникне суперечність.

Перейдемо до опису методів уточнення коріння трансцендентних рівнянь. Першим з них є метод ітерацій (послідовних наближень).

Трансцендентне рівняння

$$f(x)=0, \qquad (1.3.2.14)$$

представимо до вигляду

$$x = f_1(x), \tag{1.3.2.15}$$

де $f_1(x) = f(x) + x.$

Скориставшись графічним способом, визначимо наближене значення кореня x_O з області визначення функції $f_1(x)$ і підставимо його в праву частину рівняння (1.3.2.15). Будуватимемо послідовність чисел $\{x_n\}$, визначених за допомогою ітераційної формули

$$x_{n+1} = f_1(x_n), \quad n = 0,1,2,...$$

Послідовність чисел $\{x_n\}$ носить назву ітераційної послідовності. Якщо існує $\lim\limits_{n\to\infty} x_n = \xi,$ то переходячи в рівності (1.3.2.15) до межі і рахуючи функцію $f_1(x)$ безперервною, матимемо

$$\lim_{n\to\infty} x_{n+1} = f_1\left(\lim_{n\to\infty} x_n\right),$$

або

$$\xi = f_1(\xi).$$

З останньої рівності виходить, що ξ буде коренем трансцендентного рівняння (1.3.2.14). Ітераційний процес продовжується до тих пір, поки дотримується умова

$$|x_{n+1} - x_n| \geq \varepsilon, \tag{1.3.2.16}$$

де ε - задана погрішність обчислення кореня ξ.

Перед доведенням теореми про збіжність ітераційної послідовності зупинимося на умові Ліпшиця, яка свідчить, що функція $f(x)$ задовольняє йому, якщо існує така постійна $q<1$, що для будь-яких x_1, x_2, що належать відрізку $[a,b]$, виконується нерівність

$$\left|\frac{f(x_1) - f(x_2)}{x_1 - x_2}\right| \leq q. \tag{1.3.2.17}$$

Якщо функція (1.3.2.14) задовольняє умові (1.3.2.17), то вона є безперервною на відрізку $[a,b]$. Повідомимо аргументу x приріст Δx. Для початкового наближення визначимо приріст функції (1.3.2.14)

$$\Delta f = f(x_O + \Delta x) - f(x_O),$$

звідки, з використанням умови Ліпшиця, отримаємо підтвердження безперервності функції (1.3.2.14) на відрізку $[a,b]$

$$|\Delta f| \le \alpha |\Delta x|,$$
$$\lim_{\Delta x \to 0} \Delta f = 0.$$

Теорема про збіжність ітераційної послідовності може бути сформульована таким чином. Допустимо, що функція $f_1(x)$ визначена і диференціюєма на відрізку $[a,b]$, причому всі її значення $f_1(x) \in [a,b]$. Тоді з використанням умови Ліпшиця $\left|\frac{df_1(x)}{dx}\right| \le q < 1$ (при $a < x < b$) отримаємо, що процес ітерації $x_{n+1} = f_1(x_n)$ сходиться незалежно від початкового значення $x_O \in [a,b]$ і граничне значення $\xi = \lim_{n \to \infty} x_n$ є єдиним коренем рівняння $x = f_1(x)$ на відрізку $[a,b]$.

Доведемо це твердження. Візьмемо початкове наближення вирішення трансцендентного рівняння (1.3.2.14) x_O на відрізку $[\xi - \delta, \xi + \delta]$, яке розташоване від точки ξ на відстані не більше ніж δ ($|\xi - x_O| \le \delta$). Виконаємо ітераційний процес з використанням умови Ліпшиця і з обліком (1.3.2.15)

$$\begin{aligned} x_1 = f_1(x_O), \;\; x_1 - \xi = f_1(x_O) - f_1(\xi), \\ |x_1 - \xi| = |f_1(x_O) - f_1(\xi)| \le q|x_O - \xi| \le q\delta, \\ |x_2 - \xi| = |f_1(x_2) - f_1(\xi)| \le q|x_1 - \xi| \le q^2 |x_O - \xi| \le q^2 \delta, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ |x_n - \xi| \le q^n |x_O - \xi| \le q^n \delta. \end{aligned} \qquad (1.3.2.18)$$

З (1.3.2.18) витікає, що

$$\lim_{n \to \infty} (x_n - \xi) = 0 \Rightarrow \lim_{n \to \infty} x_n = \xi.$$

Теорема залишається правильною, якщо функція $f_1(x)$ визначена і диференціюєма на інтервалі $]-\infty, +\infty[$. Для оцінки наближення скористаємося нерівністю

$$|\xi - x_{n+1}| \le \frac{q}{1-q}|x_{n+1} - x_n|.$$

Коли $q\langle\frac{1}{2}$ то приходимо до (1.3.2.16). Остаточно

$$\xi = x_{n+1} \pm \varepsilon.$$

Перейдемо до визначення коріння трансцендентного рівняння (1.3.2.14) з використанням методу Ньютона (або метод дотичних). На рис.1.3.2.2 показана графічна схема, що реалізовує метод Ньютона.

За допомогою графічного методу визначаємо початкове наближення кореня x_O трансцендентного рівняння $f(x)=0$. Рівняння дотичної до графіка функції $f(x)$ з координатою x_O має вигляд

$$f_K = f(x_O) + f'(x_O)(x - x_O), \qquad (1.3.2.19)$$

де $f'(x_O)$ - значення похідної функції $f(x)$ в точці x_O.

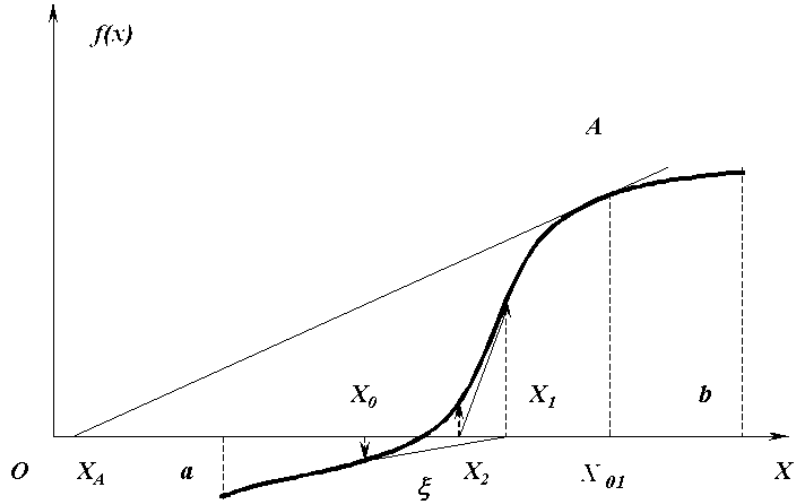


Рис.1.3.2.2. Вирішення трансцендентного рівняння методом Ньютона (дотичних)

При $x = x_1$, $f_K = 0$. Тоді з (1.3.2.19) отримаємо

$$x_1 = x_O - \frac{f(x_O)}{f'(x_O)}.$$

Продовжуючи процес побудови ітераційної послідовності $\{x_n\}$, отримаємо наступну рекурентну формулу для здійснення ітераційного процесу сходження до кореню трансцендентного рівняння

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2... \qquad (1.3.2.20)$$

Метод Ньютона, що реалізовується при допомозі (1.3.2.20), володіє високою швидкістю збіжності, проте він дуже чутливий до вибору початкового наближення x_O. На рис.1.3.2.2 показано, що вибір як початкове наближення x_{O1}(точка A на $f(x)=0$, яка розташовується далі від шуканого кореня ξ ніж x_O) приводить вже на першому кроці до випадання точки x_A з меж відрізку $[a,b]$. Це обриває процес побудови ітераційної послідовності $\{x_n\}$. Даний метод забезпечує швидку (квадратичну) збіжність якщо

$$f(x_O)f''(x_O)\rangle 0.$$

У якості x_O вибирають те значення з $[a,b]$, де знаки $f(x_O)$ і $f''(x_O)$ співпадають.

Коли $|f'(x)| \geq m\rangle 0$, $|f''(x)| \leq M$, $x \in [a,b]$ (m – найменше значення похідної $f'(x)$ в $[a,b]$; M – найбільше значення похідної $f''(x)$ в $[a,b]$), то знайдеться таке $\delta : 0\langle \delta \leq min(\xi - a, b - \xi)$, що при будь-якому виборі початкового наближення на відрізку $[\xi - \delta, \xi + \xi] \in [a,b]$ існує нескінченна ітераційна послідовність (1.3.2.20) і ця послідовність сходиться до кореню ξ трансцендентного рівняння (1.3.2.14). Дане твердження формалізує теорему про збіжність методу Ньютона. Її доказ побудований з використанням основного рекурентного співвідношення методу простих ітерацій (1.3.2.15)

$$x = f_1(x), \quad f_1(x) = x - \frac{f(x)}{f'(x)},$$
$$f_1'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}, \quad \lim_{x \to \xi} f(x) = f(\xi) = 0, \tag{1.3.2.21}$$
$$|f_1'(x)| \le \frac{M}{m^2}|f(x)|, |f(x) - f(\xi)| = |f(x)| \le \frac{m^2}{2M},$$
$$|f_1'(x)| \le q = 0{,}5.$$

Для оцінки погрішності n – го наближення x_n можна використовувати формулу

$$|\xi - x_n| \le \frac{|f(x_n)|}{m_1}, \tag{1.3.2.22}$$

де m_1 – найменше значення модуля першої похідної $|f'(x)|$ на відрізку $[a,b]$

.

Одним з недоліків методу Ньютона є необхідність обчислення похідної $f'(x)$. Для подолання цього використовується модифікований метод Ньютона, який полягає в тому, що замість обчислення похідної $f'(x)$, на кожному кроці ітерації, знаходиться її наближене значення

$$\frac{df(x)}{dx} \cong \frac{f(x_n + \Delta x) - f(x_n)}{\Delta x} = \frac{\Delta f(x_n)}{\Delta x}, \tag{1.3.2.23}$$
$$\Delta x = \varepsilon.$$

Тоді з обліком (1.3.2.23) рівняння (1.3.2.20) прийме вигляд

$$x_{n+1} = x_n - \frac{\Delta x f(x_n)}{f(x_n + \Delta x) - f(x_n)}. \tag{1.3.2.24}$$

Рекурентну формулу (1.3.2.24) використовують для побудови ітераційної послідовності $\{x_n\}$ для реалізації модифікованого методу Ньютона.

Перейдемо до розгляду методу ділення відрізку $[a,b]$ навпіл (метод дихотомії). Суть методу полягає в побудові ітераційної послідовності вкладених один в одного відрізків $[a_n, b_n]$, кінці яких представляють монотонні послідовності $\{a_n\}, \{b_n\}$, причому

$$a_n \le \xi, \quad b_n \ge \xi, \quad n = 1,2...$$

де ξ - корінь трансцендентного рівняння (1.3.2.14) на відрізку $[a,b]$.

Збіжність даного методу повільна. Проте при будь-якій ширині відрізання $[a,b]$ збіжність гарантована.

Вважатимемо, що $f(a)f(b)\langle 0$ и $f(a)\langle 0,\ f(b)\rangle 0$. Тоді візьмемо середню точку відрізку $[a,b]$

$$\xi_1 = \frac{a+b}{2}. \qquad (1.3.2.25)$$

Обчислюємо в даній крапці значення функції $f(x)$. Припустимо, що $f(\xi_1)=0$. В цьому випадку процес визначення кореня трансцендентного рівняння закінчується. Якщо $f(\xi_1)\neq 0$, то розглядаємо два відрізка $[a,\xi_1]$ и $[\xi_1,b]$. Вибираємо той з них, де виконується умова $F9a) f(\xi_1)\langle 0$ або $f(\xi_1)f(b)\langle 0$. Вибраний відрізок знову поділяємо навпіл приймаючи $a_1=a,\ b_1=\xi_1$ або $a_1=\xi_1,\ b_1=b$

$$\xi_2 = \frac{a_1+b_1}{2}. \qquad (1.3.2.26)$$

Необмежене продовження процесу ділення дозволяє отримати ітераційну послідовність вкладених один в одного відрізків $[a_n,b_n]$, причому

$$a_n \le a_{n+1}\langle b_{n+1} \le b_n. \qquad (1.3.2.27)$$

Ліві кінці відрізків утворюють ітераційну, монотонну послідовність $\{a_n\}$, яка в межі представлятиме деяку величину z_1

$$\lim_{n\to\infty} a_n = z_1, \qquad (1.3.2.28)$$

а праві кінці відрізків утворюють ітераційну, монотонну послідовність $\{b_n\}$, яка в межі представлятиме деяку величину z_2

$$\lim_{n\to\infty} b_n = z_2. \qquad (1.3.2.29)$$

Очевидно, що

$$a_n \le z_1 \le z_2 \le b_n,$$
$$z_2 - z_1 \le b_n - a_n = \frac{b-a}{2^n}. \qquad (1.3.2.30)$$

З останнього рівняння системи (1.3.2.30) отримуємо $z_2 - z_1 = 0$, т.ч. дана різниця менше будь-якого наперед заданого позитивного числа. Звідки матимемо $z_2 = z_1 = \xi$. Отже

$$f(\xi) = \lim_{n\to\infty} f(a_n) = \lim_{n\to\infty} f(b_n).$$

З обліком (1.3.2.30)

$$a_n \le \xi \le b_n.$$

Аналіз останньої нерівності показує, що величина a_n визначає шуканий корінь ξ з недоліком, а величина b_n з лишком. Помилка при цьому не перевищує довжину відрізання $b_n - a_n$ і прагне до нуля при збільшенні n за законом геометричної прогресії із знаменником 1/2. Число ітерацій N, необхідне для досягнення заданої точності ε визначається з виразу

$$N \cong \frac{ln\left(\frac{b-a}{\varepsilon}\right)}{ln\,2}.$$

Представляє інтерес метод порозрядного наближення для обчислення всього існуючого коріння трансцендентного рівняння на відрізку $[a,b]$ із заданою погрішністю ε. На рис.1.3.2.3 представлена блок-схема алгоритму для реалізації вказаного методу.

Особливість полягає в зміні величини кроку Z шляхом введення показника розрядності R (у нашому випадку приймали $R = 4$).

На початковому етапі задається крок і початок межі відрізання a. Потім виконуємо операцію привласнення $x = x + c$ і порівнюємо отриманий результат із значенням b, що визначає кінець відрізання. Далі обчислюємо значення функції (1.3.2.14) і перевіряємо умову

$$\frac{f(x)sign[f(x)]}{C} \rangle 0. \qquad (1.3.2.31)$$

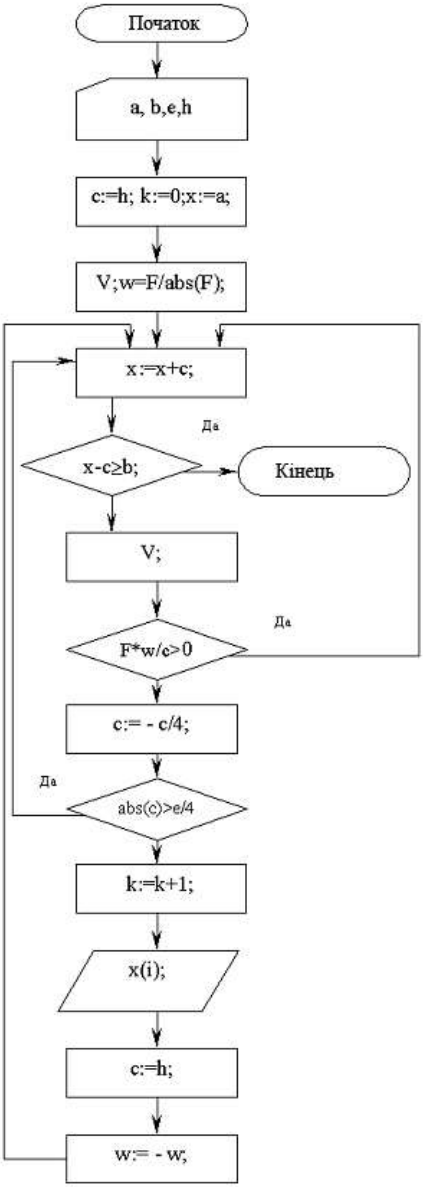


Рис.1.3.2.3. Блок-схема алгоритму для реалізації методу порозрядного наближення

На наступному етапі зменшуємо крок $C = -\frac{C}{R}$ і перевіряємо виконання умови $|C| \rangle \frac{\varepsilon}{R}$, де ε - задана погрішність обчислення кореня. Якщо умова не виконується, виводимо значення кореня на друк.

Для вирішення системи n трансцендентних рівнянь

$$f_i(x_1,...x_i) = 0, \quad i = 1,2...n, \tag{1.3.2.32}$$

скористаємося модифікованим методом Ньютона. Кожне з трансцендентних рівнянь розкладаємо в ряд Тейлор і формуємо матрицю Якобі для розрахунку приростів $f_i(x_i)$ при малій зміні змінних, замінюючи приватні похідні в матриці Якобі їх наближеними кінцево-різницевими значеннями

$$\frac{\partial f_i}{\partial x_i} \cong \frac{f_i(x_i + \varepsilon|x_i|) - f_i(x_i)}{\varepsilon|x_i|}, \tag{1.3.2.33}$$

де ε - відносна погрішність обчислення.

Матриця Якобі в розгорненому вигляді, з обліком (1.3.2.33), може бути представлена так

$$\begin{Vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{Vmatrix} \cong \tag{1.3.2.34}$$

$$\cong \begin{Vmatrix} \frac{f_1(x_1 + \varepsilon|x_1|) - f_1(x_1)}{\varepsilon|x_1|} & \cdots & \frac{f_1(x_n + \varepsilon|x_n|) - f_1(x_n)}{\varepsilon|x_n|} \\ \cdots & \cdots & \cdots \\ \frac{f_n(x_1 + \varepsilon|x_1|) - f_n(x_1)}{\varepsilon|x_1|} & \cdots & \frac{f_n(x_n + \varepsilon|x_n|) - f_n(x_n)}{\varepsilon|x_n|} \end{Vmatrix}.$$

Далі складаємо і вирішуємо систему лінійних рівнянь для малих приростів x_i

$$\begin{Vmatrix} \frac{f_1(x_1+\varepsilon|x_1|)-f_1(x_1)}{\varepsilon|x_1|} & \dots & \frac{f_1(x_n+\varepsilon|x_n|)-f_1(x_n)}{\varepsilon|x_n|} \\ \dots & \dots & \dots \\ \frac{f_n(x_1+\varepsilon|x_1|)-f_n(x_1)}{\varepsilon|x_1|} & \dots & \frac{f_n(x_n+\varepsilon|x_n|)-f_n(x_n)}{\varepsilon|x_n|} \end{Vmatrix} \cdot \begin{Vmatrix} \Delta x_1 \\ \dots \\ \Delta x_n \end{Vmatrix} = \tag{1.3.2.35}$$

$$= \begin{Vmatrix} -f_1(x_1 \dots x_n) \\ \dots \\ -f_n(x_1 \dots x_n) \end{Vmatrix}.$$

Визначені з системи (1.3.2.35) значення приростів використовуємо на наступному кроці $x_{n(i+1)} = x_{n(i)} + \Delta x_n$. Далі здійснюємо перевірку з використанням заданої величини відносної погрішності ε.

Часто при вирішенні прикладних завдань стикаємося з необхідністю визначення всього коріння полінома з дійсними коефіцієнтами

$$f(x) = x^n + a_{n+1} + \sum_{i=2}^{n} a_i x^{n-i+1}, \tag{1.3.2.36}$$

де n - ступінь полінома; a_i - дійсні числа.

Для вирішення даного завдання скористаємося схемою Горнера [14]. На початковому етапі виділяємо квадратичний тричлен і приводимо його до вигляду

$$x^2 + px + q, \tag{1.3.2.37}$$

де p, q – коефіцієнти квадратного трьохчлена.

Якщо обчислення коріння полінома (1.3.2.36) починається з найбільших по абсолютній величині значень, то беремо трьох перших членів і

$$p = a_2, \quad q = a_3, \tag{1.3.2.38}$$

а якщо обчислення коріння починаємо з найменших по абсолютній величині значень, то беремо трьох останніх членів і

$$p = \frac{a_n}{a_{n-1}}, \quad q = \frac{a_{n+1}}{a_{n-1}}. \tag{1.3.2.39}$$

Схема Горнера полягає в тому, що якщо коріння (1.3.2.37) є корінням полінома (1.3.2.36), то останній згідно теоремі Безу ділиться на (1.3.2.37)

без залишку. Пропонований метод використовувався ще математиками в середньовічному Китаї. На початку XIX століття він був «. відкритий незалежно У.Горнером (1819) і П.Руффіні (1802)».

Після виконання операції ділення (1.3.2.36) на (1.3.2.37) отриманий поліном матиме ступінь $n-2$ і з ним виконуються операції описані вище. Таким чином, вираз (1.3.2.36) буде приведений до вигляду: для випадку коли n - парне

$$f(x)=\prod_{i=1}^{\left[\frac{n}{2}\right]}\left(x^2+c_{2i}x+c_{2i+1}\right), \tag{1.3.2.40}$$

для випадку коли n - непарне

$$f(x)=\prod_{i=1}^{\left[\frac{n}{2}\right]}\left(x^2+c_{2i}x+c_{2i+1}\right)\left(x+c_{n+1}\right), \tag{1.3.2.41}$$

де c_{ij} - дійсні коефіцієнти.

Ітераційний процес визначення коріння на кожному подальшому етапі пониження ступеня полінома (1.3.2.36) проводиться по наступному алгоритму

$$\begin{aligned}
&f(x)=V(x)(x^2+px+q)=0,\\
&V(x)=x^{n-2}+b_2x^{n-3}+\ldots+b_{n-2}x+b_{n-1},\\
&b_1=1,\ \ b_2=a_2-p,\ \ b_i=a_i-pb_{i-1}-qb_{i-2},\ \ i=3\ldots n+1,\\
&c_1=1,\ \ c_2=b_2-p,\ \ c_j=b_j-pc_{j-1}-qc_{j-2},\ \ j=3\ldots n-1,\\
&c_n=-pc_{n-1}-qc_{n-2},\ \ d=c_{n-1}^2-c_nc_{n-2},\\
&\Delta p=\frac{b_nc_{n-1}-b_{n+1}c_{n-2}}{d},\ \ \Delta q=\frac{b_{n+1}c_{n-1}-b_nc_n}{d},\\
&p=p+\Delta p,\ \ q=q+\Delta q,\ \ |\Delta p|\le\varepsilon,\ \ |\Delta q|\le\varepsilon,
\end{aligned} \tag{1.3.2.42}$$

де ε - погрішність визначення коріння.

Обчислення по (1.3.2.22) проводяться поки $n\rangle 2$.

1.3.3. Диференціальні рівняння

У додатках математики до технічних наук диференціальні рівняння займають особливо важливе місце. Багато прикладних процесів з їх допомогою описуються простішим і повнішим. Вони дають можливість вирішувати багато питань загально технічних і спеціальних прикладних дисциплін: фізики, теоретичної механіки, опору матеріалів, гідравліки, теорії механізмів і машин, хімії, технології виробництв, біології, фінансово-економічних дисциплін і ін.

Відомо, що далеко не завжди вдається в замкнутому вигляді знайти вирішення диференціального рівняння. Тому розроблені методи для їх наближеного вирішення. Питання чисельного вирішення диференціальних рівнянь як звичайних, так і в приватних похідних вже багато років привертають увагу учених, проте питання про якнайкращий з методів чисельного рішення наукою поки не дозволене.

Залежно від характеру граничних умов диференціальні рівняння можна розбити на три класи: одно точкові; двох точкові і багато точкові. До першого класу відносяться звичайні диференціальні рівняння, всі граничні умови яких повинні виконуватися в одній крапці. До другого класу відносяться диференціальні рівняння, у яких граничні або початкові умови задані в двох крапках. Багато точкові граничні умови мають місце у диференціальних рівнянь порядку вище другого.

Вирішення звичайних диференціальних рівнянь з одно точковими граничними умовами приводить до покрокового завдання чисельної інтеграції.

У загальному випадку, в рамках завдання Коші (знайти таку безперервну функцію $y = y(x)$ у області $[x_O, x]$, яка задовольняє диференціальному рівнянню $\frac{dy}{dx} = f(x, y)$), ми розглядатимемо чисельну інтеграцію системи звичайних диференціальних рівнянь першого порядку

$$\begin{aligned}
&\frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots y_j, \ldots y_n),\\
&\frac{dy_2}{dx} = f_2(x, y_1, y_2, \ldots y_j, \ldots y_n),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\frac{dy_j}{dx} = f_j(x, y_1, y_2, \ldots y_j, \ldots y_n),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\frac{dy_n}{dx} = f_n(x, y_1, y_2, \ldots y_j, \ldots y_n),
\end{aligned} \qquad (1.3.3.1)$$

де x - незалежна змінна (час, переміщення, температура і ін.); y_j - залежна змінна; $j = 1 \ldots n$; n - число диференціальних рівнянь, що входять в систему (1.3.3.1).

Рішення задачі Коші зводиться до чисельної інтеграції системи диференціальних рівнянь (1.3.3.1), тобто до знаходження інтегральних залежностей

$$y_1 = y_1(x), \quad y_2 = y_2(x), \ldots y_j = y_j(x), \ldots y_n = y_n(x), \qquad (1.3.3.2)$$

за наступних початкових (граничних) умов

$$\begin{aligned}
&x = x_0,\\
&y_{10} = y_1(x_0), \quad y_{20} = y_2(x_0), \ldots y_{j0} = y_j(x_0), \ldots y_{n0} = y_n(x_0).
\end{aligned} \qquad (1.3.3.3)$$

При цьому мається на увазі, що інтегральні криві проходять через крапки, задані початковими (граничними) умовами

$$M_1(x_0, y_{10}), \quad M_2(x_0, y_{20}), \ldots M_j(x_0, y_{j0}), \ldots M_n(x_0, y_{n0}). \qquad (1.3.3.4)$$

До системи (1.3.3.1) зводиться і будь-яке диференціальне рівняння вищого порядку вигляду

$$\frac{d^n y}{dx^n} = f(x, y, \frac{dy}{dx}, \frac{d^2 y}{dx^2}, \ldots \frac{d^{n-1} y}{dx^{n-1}}). \qquad (1.3.3.5)$$

Для цього використовують метод пониження порядку диференціального рівняння (1.3.3.5) шляхом введення нових функцій

$$\frac{dy}{dx} = y_1,$$
$$\frac{dy_1}{dx} = \frac{d^2y}{dx^2} = y_2,$$
$$\cdots\cdots\cdots\cdots\cdots \qquad (1.3.3.6)$$
$$\frac{dy_{n-2}}{dx} = \frac{d^{n-1}y}{dx^{n-1}} = y_{n-1},$$
$$\frac{dy_{n-1}}{dx} = \frac{d^n y}{dx^n} = f(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \ldots \frac{d^{n-1}y}{dx^{n-1}}).$$

Методи чисельної інтеграції одного звичайного диференціального рівняння можна використовувати і для систем (1.3.3.1) (1.3.3.6). Порядок погрішності формул для чисельної інтеграції диференціальних рівнянь залежить від вибору методу побудови рекурентної формули.

Методи вирішення диференціальних рівнянь (1.3.3.1) можна розділити на: методи, що не містять в собі ніяких способів оцінки погрішності; кінцеве-різницеві методи, в яких на основі раніше набутих декількох значень функції будуються нові, які потім уточняються за допомогою самих диференціальних рівнянь.

До перших можна віднести методи Ейлера - Коші, метод трапецій, метод Рунге - Кута. До других відносяться метод Адамса, метод Крилова-Адамса.

Перейдемо до розгляду методу Ейлера - Коші. На рис.1.3.3.1 показана розрахункова схема.

Хай задано диференціальне рівняння

$$\frac{dy}{dx} = f(x, y). \qquad (1.3.3.7)$$

Необхідно знайти наближене рішення (1.3.3.7) в крапках з координатами $x_1 = x_0 + h,\ x_2 = x_0 + 2h, \ldots x_n = x_0 + nh$, де h - постійний крок; x_0 - координата початку відрізку.

Початкова умова $x = x_0,\ y_0 = y(x_0)$. Наближене значення першої похідної має вигляд

$$\frac{dy_k}{dx_k} \cong \frac{\Delta y_k}{\Delta x_k} = \frac{y_{k+1} - y_k}{h}, \qquad (1.3.3.8)$$

де $k = 0,1...n-1$.

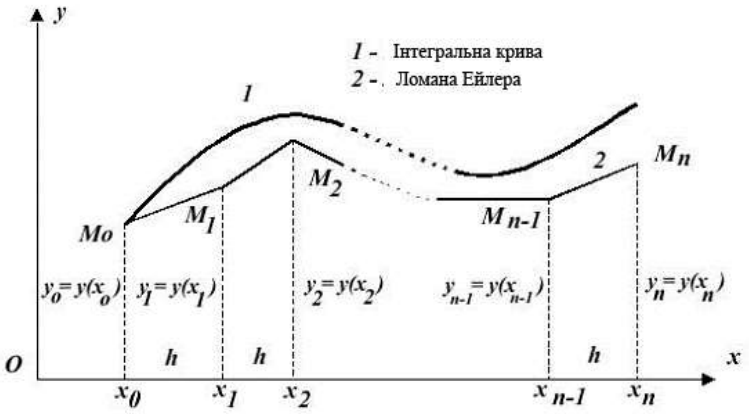


Рис.1.3.3.1. Розрахункова схема методу Ейлера-Коші

Прирівнюючи (1.3.3.7) і (1.3.3.8), отримаємо

$$\frac{y_{k+1} - y_k}{h} = f(x_k, y_k),$$

звідки

$$y_{k+1} = y_k + hf(x_k, y_k). \qquad (1.3.3.9)$$

Використовуючи рекурентну формулу (1.3.3.9) для точок $k = 0,1...n-1$ будуємо ламану Ейлера 2, яка приблизно замінює інтегральну криву 1 (см.рис.1.3.3.1). Суть методу Ейлера-Коші полягає в тому, що через початок кожного відрізку $[x_k, x_{k+1}]$ проводиться дотична до інтегральної кривої 1.

Точність методу Ейлера-Коші невелика. Погрішність методу пропорційна h^2.

Різновидом методу Ейлера-Коші є метод трапецій. Він реалізується застосуванням на кожному кроці рекурентної формули

$$y_{k+1} = y_k + \frac{h}{2}\left\{ f(x_k, y_k) + f\left[x_k + h \ , y_k + hf(x_k, y_k) \right] \right\}. \qquad (1.3.3.10)$$

Погрішність методу трапецій пропорційна h^3 і його також відносять до загальних методів Рунге - Кутта.

Перейдемо до розгляду методу Рунге-Кутта. Рекурентна формула для його реалізації має вигляд

$$y_{k+1} = y_k + hf\left[x_k + \frac{h}{2} \ , y_k + \frac{h}{2} f(x_k, y_k) \right] \ . \qquad (1.3.3.11)$$

Графічна інтерпретація даного методу показана на рис.1.3.3.2. Його суть полягає в тому, що на початковому етапі через точку M_k проводять лінію поля $f(x_k, y_k)$. Далі обчислюють координати точки M_{kc}, яка ділить відрізок $[x_k, x_{k+1}]$ навпіл.

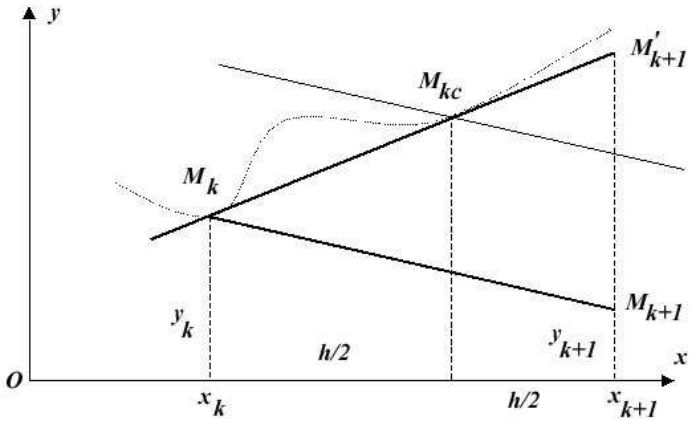


Рис.1.3.3.2. Графічна інтерпретація методу Рунге-Кутта

Його координати будуть рівні $x_k + \frac{h}{2}$, $y_k + \frac{h}{2} f(x_k, y_k)$. Лінія поля $f\left[x_k + \frac{h}{2} \ , y_k + \frac{h}{2} f(x_k, y_k) \right]$ матиме орієнтацію, відмінну від попередньої за рахунок повороту поля на інтервалі $]x_k x_{k+1}[$. Потім через точку M_k проводимо пряму, паралельну лінії нового напряму поля в точці M_{kc}. Ордината точки M_{k+1} і буде шуканою ординатою y_{k+1}.

Різновидом даного методу є метод Рунге-Кутта четвертого порядку, рекурентна формула якого має вигляд

$$\begin{aligned} a_k &= f\left[x_k + \frac{h}{2},\ \ y_k + \frac{h}{2} f(x_k, y_k)\right], \\ b_k &= f\left(x_k + \frac{h}{2},\ \ y_k + \frac{h}{2} a_k\right), \\ c_k &= f(x_k + h,\ \ y_k + b_k h), \\ y_{k+1} &= y_k + \frac{h}{6}\left[f(x_k, y_k) + 2a_k + 2b_k + c_k\right]. \end{aligned} \qquad (1.3.3.12)$$

Погрішність даного методу пропорційна h^5. Він володіє більшою стійкістю до виникнення нестійкості рішення.

Можливість забезпечувати наближену оцінку погрішності на кожному кроці інтеграції дозволяє метод Рунге-Кутта-Мерсона з автоматичною зміною кроку. Рекурентна система рівнянь має вигляд

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{6}\left[f(x_k, y_k) + 4V_{3k} + V_{4k}\right], \\ V_{1k} &= hf\left[x_k + \frac{1}{3}h,\ \ y_k + \frac{1}{3}hf(x_k, y_k)\right], \\ V_{2k} &= hf\left[x_k + \frac{1}{3}h,\ \ y_k + \frac{h}{6} f(x_k, y_k) + \frac{1}{6}V_{1k}\right], \\ V_{3k} &= hf\left[x_k + \frac{1}{2}h,\ \ y_k + \frac{h}{8} f(x_k, y_k) + \frac{3}{8}V_{2k}\right], \\ V_{4k} &= hf\left[x_k + h,\ \ y_k + \frac{h}{2} f(x_k, y_k) - \frac{3}{2}V_{2k} + 2V_{3k}\right]. \end{aligned} \qquad (1.3.3.13)$$

Погрішність обчислення y_{k+1}, з обліком (1.3.3.13), визначається по формулі

$$R_{k+1} = \frac{-2hf(x_k, y_k) + 9V_{2k} - 8V_{3k} + V_{4k}}{30}. \qquad (1.3.3.14)$$

Порівнюємо отриману погрішність R_{k+1} із заданою погрішністю E

$$|R_{k+1}| \le E,\ \ |R_{k+1}| \ge \frac{E}{30}. \qquad (1.3.3.15)$$

Якщо перша умова (1.3.3.15) не виконується, то крок h зменшується удвічі. Якщо друга умова не виконується, то крок збільшується удвічі.

Значення y_{k+1} виводиться на друк, тільки при виконанні одночасно двох умов (1.3.3.15). Погрішність чисельної інтеграції даного методу має порядок h^5.

Багатокрокові вирішення диференціальних рівнянь (кінцеве-різницеві методи) базуються на використанні даних рішення на попередніх кроках. Це дозволяє підвищити швидкість обчислення. Для реалізації кінцеве-різницевих методів при чисельному інтегруванні диференціальних рівнянь потрібно знати значення функції і її похідної в декількох точках близьких до початкової. Тут можна відзначити метод Пікара і метод розкладання. Перший метод заснований на апріорному виборі першого наближення поблизу граничної точки (x_O, y_O)

$$y = d_1(x). \tag{1.3.3.16}$$

Для (1.3.3.7), з урахуванням (1.3.3.16), отримаємо

$$\begin{gathered} y = y_O + \int_{x_O}^{x} f[x, d_1(x)]dx = d_2(x), \\ y = y_O + \int_{x_O}^{x} f[x, d_2(x)]dx = d_3(x), \\ \text{..... ....} \\ y = y_O + \int_{x_O}^{x} f[x, d_{n-1}(x)]dx = d_n(x). \end{gathered} \tag{1.3.3.17}$$

Метод Пікара поблизу граничної крапки при збіжності дає необхідну точність рішення, проте область його застосування обмежена. Це пояснюється необхідністю інтеграції (1.3.3.7) в кінцевому вигляді.

Другий метод відшукання наближеного рішення поблизу граничної крапки (x_O, y_O) пов'язаний з розкладанням в ряд Тейлора

$$\begin{gathered} y = y_O + \frac{(x-x_O)}{1!}d'(x_O) + \frac{(x-x_O)^2}{2!}d''(x_O) + ... + \\ + \frac{(x-x_O)^n}{n!}d^n(x_O) + ..., \\ y = d(x). \end{gathered} \tag{1.3.3.18}$$

Вважаючи, що

$$d'(x) = f(x,y) = \frac{dy}{dx}, \qquad (1.3.3.19)$$

диференціюємо останню рівність з урахуванням значень в граничній точці

$$\frac{d^2y}{dx^2} = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y}\frac{dy}{dx},$$

або, з урахуванням (1.3.3.19), отримаємо

$$d''(x_O) = \left[\frac{\partial f(x,y)}{\partial x}\right]_{\substack{x=x_O \\ y=y_O}} + \left[\frac{\partial f(x,y)}{\partial y}\right]_{\substack{x=x_O \\ y=y_O}} \cdot f(x_O, y_O).$$

Метод Адамса з третіми і вищими різницями реалізується за допомогою наступної рекурентної залежності

$$y_{i+1} = y_i + q_i + \frac{1}{2}\Delta q_{i-1} + \frac{5}{12}\Delta^2 q_{i-2} + \frac{3}{8}\Delta^3 q_{i-3}, \quad i = 3,4...,$$
$$q_k = hf(x_k, y_k), \quad \Delta^m q_k = h\Delta^m y'_k, \qquad (1.3.3.20)$$
$$k = 0,1,2...$$

Для реалізації (1.3.3.20) при рішенні завдання Коші необхідно визначати на початковому етапі (з використанням методу Пікара або методу розкладання) y_i, де $i = 1...3$. Дана процедура повторюється при зміні кроку інтеграції.

Враховуючи чисельну нестійкість рішення, складність програмної реалізації багатокрокові методи чисельного інтегрування диференціальних рівнянь використовуються не часто.

Приклад № 1. Визначення форми балона.

Прядіння і перемотування пряжі супроводжуються переміщенням нитки. Специфіка даних технологічних процесів приводить до виникнення криволінійної ділянки нитки між нитко спрямовувачем (див. рис.1.3.3.3) ***A*** і бігунком ***B.*** Дана ділянка має двояку кривизну. При цьому нитка бере участь в двох рухах: переносному навколо осі ***OZ*** і відносному уздовж своєї квазістатичної форми рівноваги. На нитку діятимуть сили опору

повітря, сила тяжіння, переносна і кориолісові сили інерції. В рамках справжнього завдання вважатимемо відносну лінійну швидкість і переносну кутову швидкість постійною.

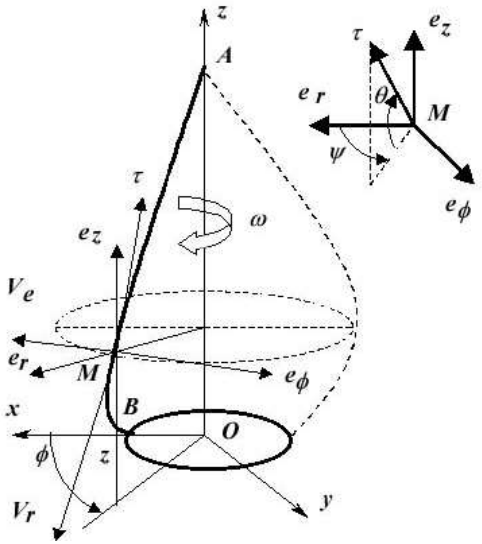


Рис.1.3.3.3. Розрахункова схема для визначення форми балонуючої нитки

Для визначення форми балона введемо три координатні системи. Декартову $OXYZ$, циліндричну з одиничними ортами e_r, e_φ, e_Z і сферичну з одиничними ортами $\cos\theta\cos\psi, \cos\theta\sin\psi, \sin\theta$. Кут θ утворює орт дотичної осі τ в даній точці M з горизонтальною площиною циліндричної координатної системи. Кут ψ утворює проекція орта дотичної на горизонтальну площину циліндричної координатної системи з одиничним ортом e_r.

Введення сферичної координатної системи дозволяє спростити початкову систему диференціальних рівнянь. Дані координати повинні задовольняти нелінійному не голономному зв'язку

$$\left(\frac{dr}{ds}\right)^2 + r^2\left(\frac{d\varphi}{ds}\right)^2 + \left(\frac{dz}{ds}\right)^2 = 1,$$

$$\frac{dr}{ds} = \cos\theta\cos\psi,$$

$$r\frac{d\varphi}{ds} = \cos\theta\sin\psi, \qquad (1.3.3.21)$$

$$\frac{dz}{ds} = \sin\theta,$$

де s - дугова координата.

Основна система диференціальних рівнянь рівноваги елементарного відрізання нитки ds у проекціях на осі циліндрової координатної системи має вигляд

$$\begin{gathered}\frac{d}{ds}\left(P\frac{dr}{ds}\right)-Pr\left(\frac{d\varphi}{ds}\right)^2+F_r=0,\\ \frac{d}{ds}\left(Pr^2\frac{d\varphi}{ds}\right)+rF_\varphi=0,\\ \frac{d}{ds}\left(P\frac{dz}{ds}\right)+F_z=0,\end{gathered} \tag{1.3.3.22}$$

де P - натяг нитки; F_r, F_φ, F_Z - проекції активно заданих сил на осі циліндрової координатної системи.

З другого рівняння системи (1.3.3.21) отримаємо диференціального оператора

$$\frac{d}{ds}=\cos\theta\cos\psi\frac{d}{dr}. \tag{1.3.3.23}$$

Використовуючи його, перетворимо третє і четверте диференціальні рівняння системи (1.3.3.21)

$$\begin{gathered}\frac{d\varphi}{dr}=\frac{tg\psi}{r},\ \frac{dz}{dr}=\frac{tg\theta}{\cos\psi},\\ \frac{ds}{dr}=\frac{1}{\cos\theta\cos\psi}.\end{gathered} \tag{1.3.3.24}$$

Використовуючи (1.3.3.23) перетворимо систему диференціальних рівнянь (1.3.3.21)

$$\begin{gathered}\cos\theta\cos\psi\left[\cos\theta\cos\psi\frac{dP}{dr}+P\frac{d}{dr}\left(\cos\theta\cos\psi\right)\right]-\frac{P}{r}\cos^2\theta\sin^2\psi+F_r=0,\\ \cos\theta\cos\psi\left[r\cos\theta\sin\psi\frac{dP}{dr}+P\frac{d}{dr}\left(r\cos\theta\sin\psi\right)\right]+rF_\varphi=0,\\ \cos\theta\cos\psi\left[\sin\theta\frac{dP}{dr}+P\frac{d}{dr}\left(\sin\theta\right)\right]+F_Z=0,\end{gathered} \tag{1.3.3.25}$$

Помножимо перше рівняння (1.3.3.25) на $\cos\theta\cos\psi$, друге на $\frac{\cos\theta\sin\psi}{r}$, а третє на $\sin\theta$ і складемо їх [9]

$$\frac{dP}{dr}=-\left(F_r+F_\varphi tg\psi+F_Z\frac{tg\theta}{\cos\psi}\right). \tag{1.3.3.26}$$

Перетворимо перші два диференціальні рівняння системи (1.3.3.25) до вигляду [20]

$$\cos\theta\cos\psi\left[-P\cos\theta\sin\psi\frac{d\psi}{dr}+\cos\psi\frac{d}{dr}\left(P\cos\theta\right)\right]-\frac{P}{r}\cos^2\theta\sin^2\psi+F_r=0,$$

$$\cos\theta\cos\psi\left[Pr\cos\theta\cos\psi\frac{d\psi}{dr}+r\sin\psi\frac{d}{dr}\left(P\cos\theta\right)+P\cos\theta\sin\psi\right]+rF_\varphi=0.$$

Помножимо перше з отриманих рівнянь на $-r\sin\psi$, а друге на $\cos\psi$. Складемо результати, тоді

$$\frac{d\psi}{dr}=\frac{F_r\sin\psi-F_\varphi\cos\psi}{P\cos^2\theta\cos\psi}-\frac{tg\psi}{r}. \qquad (1.3.3.27)$$

Перетворимо третє рівняння системи (1.3.3.25) з урахуванням (1.3.3.26) до вигляду

$$\frac{d\theta}{dr}=\frac{tg\theta}{P}\left(F_r+F_\varphi tg\psi-\frac{F_Z}{tg\theta\cos\psi}\right). \qquad (1.3.3.28)$$

Рівняння (1.3.3.24) (1.3.3.26) (1.3.3.27).(1.3.3.28) утворюють систему п'яти диференціальних рівнянь першого порядку. Залишається визначити проекції F_r, F_φ, F_Z активно заданих сил на осі циліндрової координатної системи. Головний вектор зовнішніх сил буде рівний

$$\begin{gathered}\vec{F}=F_r\vec{e}_r+F_\varphi\vec{e}_\varphi+F_Z\vec{e}_Z,\\ F_r=\mu\omega^2r-2\mu\omega u\cos\theta\sin\psi-\lambda_2r^2\Phi_n(\chi)ctg\chi\cos\theta\cos\psi,\\ F_\varphi=2\mu\omega u\cos\theta\cos\psi+\lambda_2r^2\Phi_n(\chi)\frac{1}{\sin\chi}-\lambda_2r^2\Phi_n(\chi)ctg\chi\cos\theta\sin\psi,\\ F_Z=-q-\lambda_2r^2\Phi_n(\chi)ctg\chi\sin\theta,\\ \Phi_n(\chi)=a\sin^2\chi,\\ \lambda_2=C\rho\frac{\omega^2}{2}d,\end{gathered} \qquad (1.3.3.29)$$

де μ- лінійна щільність нитки; ω - переносна кутова швидкість; u - лінійна відносна швидкість; χ - кут атаки між ортом дотичної τ і вектором лінійної переносної швидкості елементу нитки; C, a- деякі постійні; q - питома вага нитки; ρ - щільність повітря; d - діаметр нитки.

Підставляючи отримані результати з (1.3.3.29) в (1.3.3.24) (1.3.3.26) (1.3.3.27).(1.3.3.28), отримаємо наступну систему диференціальних рівнянь

$$\frac{dP}{dr} = q\frac{dz}{dr} - \mu\omega^2 r,$$

$$\frac{d\theta}{dr} = \frac{1}{P}\left[tg\theta\left(\mu\omega^2 r + \lambda_2 r^2 a\sin\chi tg\psi\right) + \frac{q}{\cos\psi}\right],$$

$$\frac{d\psi}{dr} = \frac{\mu\omega^2 r\sin\psi - \lambda_2 r^2 a\sin\chi - 2\mu\omega u\cos\theta}{P\cos^2\theta\cos\psi} - \frac{tg\psi}{r}, \qquad (1.3.3.30)$$

$$\frac{d\varphi}{dr} = \frac{tg\psi}{r},$$

$$\frac{dz}{dr} = \frac{tg\theta}{\cos\psi}.$$

Отриману систему диференціальних рівнянь необхідно інтегрувати з використанням чисельних методів. При розрахунках приймаємо $\lambda_2 a\sin\chi = 1$. При інтеграції необхідно використовувати наступні початкові умови: при $r = r_B = 50мм,\ \varphi = \varphi_O = 0,\ z = z_O = 0$.

При інтегруванні системи диференціальних рівнянь (1.3.3.30) використовувався метод Рунге-Кутта четвертого порядку. Розрахунки проводилися для бавовняної пряжі 25 текс при наступних значеннях: $\mu = 25\cdot 10^{-6} кг/м,\ q = 25\cdot 10^{-5} н/м,\ \omega = 520с^{-1}(5000об/мин),\ u = 20м/с$.

Інтегрування виконували за наступних початкових умов: $P = P_O = 0{,}2H,\ \theta = \theta_O = 1{,}2рад,\ \psi = \psi_O = 0{,}7рад$.

На першому етапі виконували інтеграцію, коли виконується умова $\psi \le 1{,}57рад$ (відповідає максимальному значенню r_{max}). На другому етапі виконуємо інтеграцію, коли виконується умова $r \ge 0$.

1.3.4. Чисельне інтегрування

Чисельну інтеграцію використовують тоді, коли знаходження первісною скрутно або неможливо. Процес чисельної інтеграції полягає в інтерполяції підінтегральної функції $f(x)$ на відрізку $[a,b]$ відповідним поліномом, для якого певний інтеграл обчислюється по формулах

чисельної інтеграції. Вибір виду полінома визначається властивістю класу функцій, для наближення якого призначаються інтерполяційні формули. Для наближення періодичних функцій (з періодом 2π) використовують тригонометричну систему функцій. Для наближення обмежених або зростаючих функцій на піввісь $[0,\infty)$ ефективно використовувати систему раціональних або показових функцій, які враховують поведінку функцій, що наближаються, на нескінченності.

Широкого поширення набула інтерполяція алгебри з використанням інтерполяційних многочленів (Лагранжа, Ньютона, Гауса, Бесселя і ін.). На практиці найчастіше використовуються параболічні або кубічні поліноміальні сплайни, які на кожному з відрізків $[x_{n-1}, x_n]$ представляють многочлени 2-го і 3-го ступеня, що належать класу двічі функцій, що безперервно диференціюються.

Хай задана підінтегральна функція $y = f(x)$ і відомі її значення в точках $x_0 = a,\ x_1, x_2, ... x_n = b$, причому $y_i = f(x_i)$, $(i = 0,1,2,...n)$. Необхідно визначити функцію, що наближає $g(x)$ в точках $x_0, x_1, x_2, ... x_n$, щоб виконувалася наближена рівність

$$\int_a^b f(x)dx \cong \int_a^b g(x)dx. \qquad (1.3.4.1)$$

Інтерполяційний поліном Лагранжа для функції $f(x)$ буде мати вигляд

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{j \neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} == \sum_{i=0}^{n} f(x_i) \frac{\prod_{j=0}^{n} (x - x_j)}{(x - x_i) \prod_{j \neq i}^{n} (x_i - x_j)}. \qquad (1.3.4.2)$$

Інтерполяційний поліном Ньютона з розділеними різницями матиме вигляд

$$L_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + ... + f(x_0, x_1, x_2, ... x_n) \prod_{j-0}^{n-1} (x - x_j), \qquad (1.3.4.3)$$

де розділені різниці першого, другого, $k-1$ порядку визначатимуться по формулах

$$f(x_i,x_j)=\frac{f(x_j)-f(x_i)}{x_j-x_i},$$

$$f(x_i,x_j,x_k)=\frac{f(x_j,x_k)-f(x_i,x_k)}{x_k-x_i},$$

$$f(x_O,x_1,...x_k)=\prod_{j=0}^{k}\left[\frac{f(x_j)}{\prod\limits_{\substack{j\neq i\\ i=0}}^{k}(x_j-x_i)}\right].$$

Використовуючи вирази для інтерполяційних поліномів (1.3.4.2), (1.3.4.3), отримаємо з (1.3.4.1)

$$\int_a^b f(x)=\int_a^b L_n(x)dx+R_n, \qquad (1.3.4.4)$$

де R_n - погрішність квадратурної формули.

Величина R_n на класі F підінтегральних функцій обчислюється за формулою

$$R_n(F)=\sup_{f\in F}\left|\sum_{k=0}^{n}C_K f(x_k)-\int_{\Omega} f(x)\omega(x)dx.\right| \qquad (1.3.1.3.5)$$

У (1.3.1.3.5) вираз, що стоїть в модульних дужках визначає погрішність квадратури при обчисленні інтеграла від заданої функції $f(x)$.

Наближене значення інтеграла може бути визначене з (1.3.4.4) по формулі

$$\int_a^b f(x)dx=\int_{x_O}^{x_n} f(x)dx\cong\sum_{i=O}^{n}A_i f(x_i),$$

$$A_i=(x_n-x_O)H_i,$$

$$H_i=\frac{(-1)^{n-i}}{ni!(n-i)!}\int_O^n\frac{u(u-1)(u-2)...(u-n)}{u-i}du, \qquad (1.3.1.3.6)$$

$$u=\frac{(x-x_O)n}{(x_n-x_O)},\quad i=0,1,2,...n.$$

Для певного значення n коефіцієнти Лагранжа H_i приймають певні значення. При $n=2$ (функція, що наближає, є поліномом другого ступеня) набудемо наступних значень коефіцієнтів Лагранжа

$$H_O = \frac{1}{4}\int_0^2 (u-1)(u-2)du = \frac{1}{6},$$
$$H_1 = -\frac{1}{2}\int_0^2 u(u-2)du = \frac{2}{3}, \qquad (1.3.4.7)$$
$$H_2 = \frac{1}{4}\int_0^2 u(u-1)du = \frac{1}{6}.$$

Підставляємо (1.3.4.7) в (1.3.1.3.6), тоді

$$\int_{x_O}^{x_2} f(x)dx \cong \frac{(x_2 - x_O)}{6}\left[f(x_O) + 4f(x_1) + f(x_2)\right]. \qquad (1.3.4.8)$$

Розіб'ємо відрізок $[a,b]$ на парне число n ділянок $[x_O, x_2], [x_2, x_4], \ldots [x_{n-2}, x_n]$ довжини $(b-a)/n = h$. Використовуючи (1.3.4.8) і (1.3.1.3.6) отримаємо

$$\int_a^b f(x)dx \cong \frac{h}{3}\left[f(x_O) + 4f(x_1) + f(x_2)\right] + \frac{h}{3}\left[f(x_2) + 4f(x_3) + f(x_4)\right] +$$
$$+ \ldots + \frac{h}{3}\left[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right].$$

Згрупуємо члени і отримаємо відому формулу Сімпсона [14]

$$\int_a^b f(x)dx \cong \frac{b-a}{3n}\{f(a) + f(b) + 4\left[f(x_1) + f(x_3) + \ldots + f(x_{n-1})\right] + \qquad (1.3.4.9)$$
$$2\left[f(x_2) + f(x_4) + \ldots + f(x_{n-2})\right]\}.$$

На рис.4.1.3.1 представлена графічна схема, що ілюструє інтерполяцію підінтегральної функції $f(x)$ і обчислення певного інтеграла на відрізку $[a,b]$ з використанням полінома другого ступеня по формулі Сімпсона.

Погрішність квадратурної формули Сімпсона, за умови безперервності четвертої похідної функції $f(x)$ на відрізку $[a,b]$визначається з виразу

$$R_n = \frac{(b-a)h^4}{180}\left|y^{IV}(x_*)\right|,$$

де x_* - точка, що належить відрізку $[a,b]$.

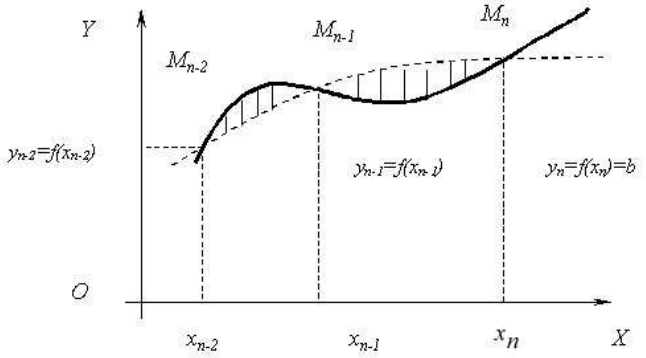


Рис. 1.3.4.1. Розрахункова схема інтерполяції підінтегральної функції по формулі Сімпсона

При $n=1$ (функція, що наближає, є лінійною) з (1.3.1.3.6) набудемо наступних значень коефіцієнтів Лагранжа

$$H_O = -\int_0^1 (u-1)du = \frac{1}{2}$$
$$H_1 = \int_0^1 udu = \frac{1}{2}. \qquad (1.3.4.10)$$

Підставляємо (1.3.4.10) в (1.3.1.3.6), тоді

$$\int_{x_o}^{x_1} f(x)dx \cong \frac{(x_1 - x_O)}{2}\left[f(x_O) + f(x_1)\right]. \qquad (1.3.4.11)$$

Розіб'ємо відрізок $[a,b]$ на n рівних частин $[x_O = a, x_1], [x_1, x_2], [x_2, x_3], \ldots [x_{n-1}, x_n = b]$ тоді

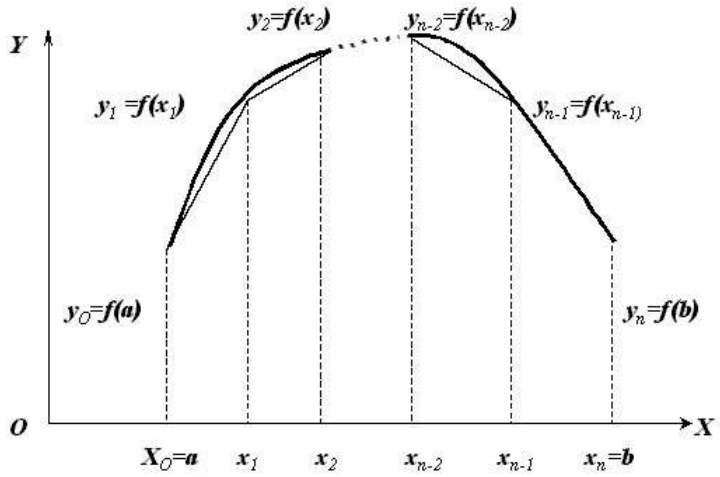
$$x_1 - x_O = \frac{b-a}{n} = h. \qquad (1.3.4.12)$$

Використовуючи (1.3.4.11).(1.3.4.12) і (1.3.1.3.6) отримаємо з урахуванням суміжних точок відрізку

$$\int_a^b f(x)dx \cong \frac{b-a}{n}\left[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{1}{2}f(b)\right]. \qquad (1.3.4.13)$$

Формула (4.1.3.13) є формулою трапецій для наближеного обчислення певного інтеграла (1.3.4.1).

Геометричний сенс (1.3.4.13) (див. рис. 1.3.4.2) полягає в тому, що підінтегральна функція $f(x)$ замінюється на кожному інтервалі деякою лінійною функцією, що наближає. На кожному інтервалі $[a,x_1],[x_1,x_2],[x_2,x_3],...[x_{n-1},b]$ визначається площа трапеції. Висота для всіх трапецій, за умови постійності кроку розбиття, дорівнюватиме $h=(b-a)/n$.



Ріс. 1.3.4.2. Геометрична інтерпретація формули трапеції

Погрішність квадратурної формули трапецій, за умови безперервності другої похідної функції $f(x)$ на відрізку $[a,b]$ визначається з виразу

$$R_n=\frac{(b-a)^3}{12n^2}\left|f^{II}(x_*)\right|. \qquad (1.3.4.14)$$

При $n=0$ (функція, що наближає, є лінійній, паралельній осі x) з (1.3.1.3.6) набудемо значення коефіцієнта Лагранжа

$$H_0=1. \qquad (1.3.4.15)$$

Підставляємо (1.3.4.15) в (1.3.1.3.6), тоді для відрізання $[x_0,x_1]$ отримаємо

$$\int_{x_0}^{x_1} f(x)dx \cong (x_1-x_0)f(x_0). \qquad (1.3.4.16)$$

Розіб'ємо відрізок $[a,b]$ на n рівних частин $[x_O=a,x_1],[x_1,x_2],[x_2,x_3],...[x_{n-2},x_{n-1}],[x_{n-1},x_n=b]$. Середини даних відрізків визначаються з виразів $\xi_1=a+\frac{h}{2}$, $\xi_2=x_1+\frac{h}{2},...\xi_{n-1}=x_{n-2}+\frac{h}{2}$, $\xi_n=x_{n-1}+\frac{h}{2}$. Використовуючи (1.3.4.12), (1.3.4.16) і (1.3.1.3.6) отримаємо

$$\int_a^b f(x)dx\cong\left[f(\xi_1)+f(\xi_2)+f(\xi_3)+...+f(\xi_{n-1})+f(\xi_n)\right]\frac{b-a}{n}. \qquad (1.3.4.17)$$

Вираз (1.3.4.17) є формулою прямокутників для наближеного обчислення певного інтеграла (1.3.4.1).

Геометричний сенс (1.3.4.17) (див. рис. 1.3.4.3) полягає в тому, що площа криволінійної трапеції замінюється площею фігури, що складається з прямокутників. Всі прямокутники мають однакову підставу $(b-a)/n$ а їх висоти визначаються значеннями функції $f(x)$ у середніх точках $\xi_1,\xi_2,...\xi_{n-1},\xi_n$ відрізків $[x_O=a,x_1],[x_1,x_2],[x_2,x_3],...[x_{n-2},x_{n-1}],[x_{n-1},x_n=b]$.

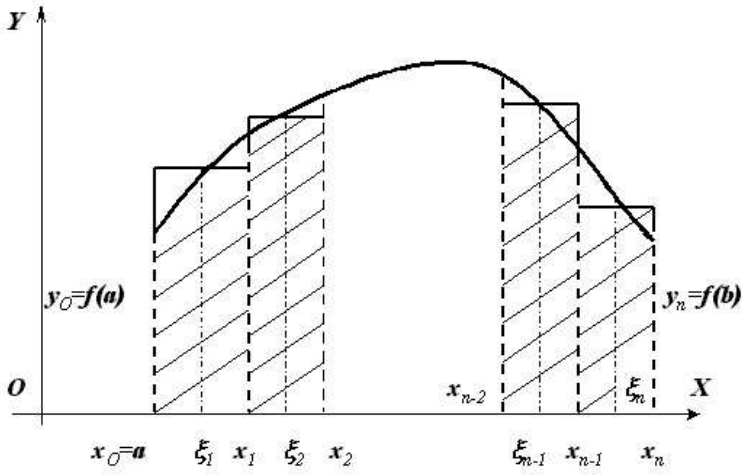


Рис. 1.3.4.3. Геометрична інтерпретація формули прямокутників

Погрішність квадратурної формули прямокутників, за умови безперервності другої похідної функції $f(x)$ на відрізку $[a,b]$визначається з виразу

$$R_n = \frac{(b-a)^3}{24n^2}\left|f^{II}(x_*)\right|. \qquad (1.3.4.18)$$

Аналіз залежностей (1.3.4.18) і (1.3.4.14) дозволяє стверджувати, що формули прямокутників і трапецій володіють приблизно однаковою точністю. Вибір тієї або іншої формули залежить від виду підінтегральної формули, обчислювальних можливостей ЕОМ.

1.3.5. Інтерполяція, екстраполяція і апроксимація

Як наголошувалося в попередньому підрозділі, при розгляді чисельної інтеграції питання, зв'язані з наближенням функцій, виникають при чисельній інтеграції, вирішенні диференціальних рівнянь, обробці експериментальних даних. Зокрема, наближення довільного контуру швейних лекал, деталей взуття простими функціями.

По характеру наближення методи наближення розділяють на інтерполяцію, екстраполяцію і апроксимацію.

Спосіб наближення функції, коли функція, що наближає $g(x)$ у заданих точках співпадає по значеннях із заданою $f(x)$називається інтерполяцією. Хай задана деяка сукупність крапок $x_1, x_2, x_3, ..., x_n$. Інтерполяція має на увазі виконання наступної рівності

$$f(x_i) = g(x_i), \quad i = 1,2,...n. \qquad (1.3.5.1)$$

Метод інтерполяції іноді називають точним методом. При екстраполюванні використовується інформація про поведінку функції в деякому кінцевому наборі крапок (у вузлах інтерполяції) і робиться припущення про подальше її поведінці. Даний метод іноді називають змішаним методом.

Під апроксимацією мається на увазі заміна заданій функції $f(x)$ деякою функцією $f_1(x)$ яка добре відображає властивості досліджуваного процесу, і різниця між значеннями даних функцій в деяких крапках з області їх визначення була б мінімальною

$$v_i = |f_1(x_i) - f(x_i)| = min, \quad i = 1,2,...n, \tag{1.3.5.2}$$

де v_1 - різниця між значенням заданої функції $f(x_i)$ і функції, що наближає $f_1(x_i)$ у деякій крапці x_i.

Іноді даний метод називають наближеним методом аналітичного опису дискретно заданої функції (по фіксованих значеннях у вузлах інтерполяції).

При обробці експериментальних даних вид функції, що наближає, визначають з вимоги мінімізації суми

$$S = min\left\{\sum_{k=1}^{n} \alpha_k [f(x_k) - f_1(x_k)]^2\right\}, \tag{1.3.5.3}$$

де α_k - задані числа.

Такий підхід до визначення функції, що наближає $f_1(x)$ називається інтерполяцією по методу найменших квадратів.

Найбільш поширений метод лінійної інтерполяції, коли функція (1.3.5.1), що наближає, представляється інтерполяційним многочленом

$$g(x) = \sum_{i=1}^{n} a_i \varphi_i(x), \tag{1.3.5.4}$$

де a_i - коефіцієнти многочлена (1.3.5.4), визначувані з умови збіги заданої функції $f(x)$ і функції, що наближає $g(x)$ у крапках $x_1, x_2, x_3, ...x_n$; $\varphi_i(x)$ – деякі фіксовані функції.

Обчислення коефіцієнтів многочлена (1.3.5.4) можна проводити по методу Ейткена шляхом послідовного застосування формул лінійної

інтерполяції при довільному розташуванні вузлів без явної побудови інтерполяційного полінома по формулах

$$P_{i,i+1}(x)=\frac{1}{x_{i+1}-x_i}\begin{vmatrix} x-x_i & f(x_i) \\ x-x_{i+1} & f(x_{i+1}) \end{vmatrix}, \quad i=1,2...n-1,$$
$$P_{j-1,j,j+1}(x)=\frac{1}{x_{j+1}-x_{j-1}}\begin{vmatrix} x-x_{j-1} & P_{j-1,j}(x) \\ x-x_{j+1} & P_{j,j+1}(x) \end{vmatrix}, \quad j=2,...n-1. \qquad (1.3.5.5)$$

Хай у нас задані значення $f(x_1)$=10, $f(x_2)$= 4 $f(x_3)$=2, $f(x_4)$=8, при x_1=0, x_2=2, x_3=3, x_4=6. Використовуючи (1.3.5.5) по методу Ейткена за допомогою функціональних визначників другого порядку побудуємо інтерполяційний поліном

$$P_{1,2}(x)=\frac{1}{2}\begin{vmatrix} x & 10 \\ x-2 & 4 \end{vmatrix}=\frac{1}{2}(4x-10x+20)=10-3x,$$
$$P_{2,3}(x)=\frac{1}{1}\begin{vmatrix} x-2 & 4 \\ x-3 & 2 \end{vmatrix}=8-2x,$$
$$P_{3,4}(x)=\frac{1}{3}\begin{vmatrix} x-3 & 2 \\ x-6 & 8 \end{vmatrix}=\frac{1}{3}(8x-24-2x+12)=2x-4,$$
$$P_{1,2,3}(x)=\frac{1}{3}\begin{vmatrix} x & 10-3x \\ x-3 & 8-2x \end{vmatrix}=\frac{1}{3}(8x-2x^2+3x^2-19x+30)=\frac{1}{3}(x^2-11x+30),$$
$$P_{2,3,4}(x)=\frac{1}{4}\begin{vmatrix} x-2 & 8-2x \\ x-6 & 2x-4 \end{vmatrix}=\frac{1}{4}(4x^2-28x+56)=x^2-7x+14,$$
$$P_{1,2,3,4}(x)=\frac{1}{6}\begin{vmatrix} x & \frac{1}{3}(x^2-11x+30) \\ x-6 & x^2-7x+14 \end{vmatrix}=\frac{1}{6}\left[x^3-7x^2+14x-\frac{(x-6)}{3}(x^2-11x+30)\right].$$

Для перевірки правильності обчислення підставимо в останнє значення x_1 =0 і x_2=2, отримаємо $P_{1,2,3,4}(0)$=10, $P_{1,2,3,4}(2)$= 4.

Як наголошувалося в підрозділі 1.3.4, найчастіше при інтерполяції використовують інтерполяційні поліноми Лагранжа і Ньютона. Для періодичних функцій $f(x)$ з періодом $b-a$ ($x_1=a, x_n=b$) функція, що наближає $g(x)$ може бути представлена тригонометричним поліномом

$$LT_n(x)=A+\sum_{k=1}^{n}(a_k\cos kx+b_k\sin kx), \qquad (1.3.5.6)$$

де A, a_k, b_k - коефіцієнти тригонометричного полінома (1.3.5.6). Завжди можна підібрати $2n+1$ коефіцієнтів A, a_k, b_k полінома n-го порядку так, щоб значення функції, що наближає, були рівні значенням $f(x_k)$ у $2n+1$ наперед заданих точках x_k проміжку $[a=0,\ b=2\pi]$ 2π- періодичній функції.

Використовуючи приведений вище вираз інтерполяційного полінома Лагранжа, представимо тригонометричні поліноми на відрізку $[-\pi]$ для парної функції $f(x)$

$$LT_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{\cos x - \cos x_j}{\cos x_i - \cos x_j},$$

для непарної функції $f(x)$

$$LT_n(x) = \sum_{i=1}^{n} f(x_i) \frac{\sin x}{\sin x_i} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{\cos x - \cos x_j}{\cos x_i - \cos x_j}.$$

Враховуючи складність побудови інтерполяційних тригонометричних многочленів відрізок $[a=0,\ b=2\pi]$ розбивається на вузли, віддалені один від одного на рівні відстані $h = 2\pi / (2n+1)$.

У звичайної полі номінальної інтерполяції зміна $f(x)$ поблизу якого-небудь вузла робить вплив на зміну $f(x)$ по всій кривій, що якісно невірно описує фізичні явища лежачі в основі не лінійності інтерпольованої функції. Нерідко із збільшенням числа вузлів погрішність такої інтерполяції не тільки не зменшується, але і починає рости. Апаратом наближення функцій, позбавленим цих недоліків, є сплайни – функції, що складаються з різних многочленів, які наближають функцію на фіксованих ділянках відрізання $[a,b]$. Сплайн - інтерполяція відноситься до багато інтервальної інтерполяції і забезпечує не тільки рівність $f(x_i) = g(x_i)$ у

вузлах інтерполяції, але і безперервність заданого числа похідних на межах конкретних інтервалів. Гідністю локально заданих сплайнів є опис властивостей функції $f(x)$ на кожному окремому інтервалі незалежно від її властивостей на інших інтервалах.

На практиці ширше уживаються сплайни невисокого ступеня – параболічні і кубічні. Вони представляють функцію інтерполяції у вигляді полінома другого або третього ступеня.

Сплайн-функція $S_m(\Delta n; x)$ ($m = 2, 3$) визначена на відрізку $[a,b]$ і співпадає на відрізках $[x_i, x_{i+1}]$ з деякими многочленами алгебри ступеня не вище m. Сплайн-функція має на відрізку $[a,b]$ безперервну $m-1$ похідну. Сітка $\Delta n : a = x_0 < x_1 < ... < x_n = b$ розбиває відрізок $[a,b]$ на часткові відрізки. Якщо сплайн-функція має на відрізку $[a,b]$ безперервну $m-k$-ю похідну ($k \geq 1$), а $m-k+1$-я похідна у вузлах сплайна має розрив, то говорять, що сплайн має дефект порядку k.

Загальний вираз для сплайн - функції має вигляд

$$S_m(\Delta_n; x) = P_{m-1}(x) + \sum_{k=0}^{n-1} c_k (x - x_k)_+^m, \qquad (1.3.5.7)$$

де c_k - дійсні числа $P_{m-1}(x)$ - многочлен алгебри ступеня не вищий $m-1$; $(x - x_k)_+^m = [max(0, x - x_k)]^m$.

Для параболічного сплайна потрібно дві додаткові краєві умови

$$\begin{aligned} S_2'(a) &= a_n, \; S_2'(b) = b_n, \\ S_2''(a) &= A_n, S_2''(b) = B_n. \end{aligned} \qquad (1.3.5.8)$$

Якщо функція $f(x)$ має першу і другу похідні на відрізку $[a,b]$то значення системи (1.3.5.8) можуть бути визначені з наступної системи рівнянь

$$\begin{aligned} a_n &= f'(a), \; b_n = f'(b), \\ A_n &= f''(a), \; B_n = f''(b). \end{aligned} \qquad (1.3.5.9)$$

Для параболічного сплайна отримаємо

$$S_2(x) = f(x_i) + m_i(x - x_i) + c_i(x - x_i)^2,$$
$$c_i = \frac{m_{i+1} - m_i}{2h_i} - \frac{f(x_{i+1}) - f(x_i)}{h_i(\bar{h}_i - h_i)} + \frac{m_{i+1} + m_i}{2(\bar{h}_i - h_i)}, \qquad (1.3.5.10)$$
$$h_i = x_{i+1} - x_i,\ \bar{h}_i = x_{i+1} - \bar{x}_{i+1},\ m_i = S_2'(x_i),$$
$$i = 0,...n,$$

де $\bar{x}_i$ - координата вузла сплайна або точка можливого розриву другої похідної; x_i - координата інтерполяції.

На практиці частіше розглядаються випадки, коли координати вузлів сплайна розташовуються посередині, між вузлами інтерполяції

$$\bar{x}_i = x_i + \frac{x_{i+1} - x_i}{2},\quad i = 0,...n.$$

Для кубічного сплайна на часткових відрізках $[x_i, x_{i+1}]$ загальна формула, з обліком (1.3.5.7), має вигляд

$$S_3(x) = \frac{1}{6h_i}\left[m_i(x_{i+1} - x)^3 + m_{i+1}(x - x_i)^3\right] +$$
$$+\frac{1}{h_i}\left\{\left[f(x_i) - \frac{m_i h_i^2}{6}\right](x_{i+1} - x) + \left[f(x_{i+1}) - \frac{m_{i+1}h_i^2}{6}\right](x - x_i)\right\}, \qquad (1.3.5.11)$$
$$h_i = x_{i+1} - x_i,\ m_i = f''(x_i),\ i = 1,2,...n,$$

де n - число вузлів.

При інтерполяції $f(x_i) = S_3(x_i)$ отримаємо систему лінійних рівнянь для визначення m_i

$$h_i m_i + 2(h_i + h_{i+1})m_{i+1} + h_{i+1}m_{i+2} = 6\left[\frac{f(x_{i+2}) - f(x_{i+1})}{h_{i+1}} - \frac{f(x_{i+1}) - f(x_i)}{h_i}\right], \qquad (1.3.5.12)$$
$$i = 1,2,...n.$$

Систему лінійних рівнянь (1.3.5.12) необхідно доповнити граничними умовами

$$S_3'(a) = a_n,\ S_3'(b) = b_n,$$
$$S_3''(a) = A_n, S_3''(b) = B_n. \qquad (1.3.5.13)$$

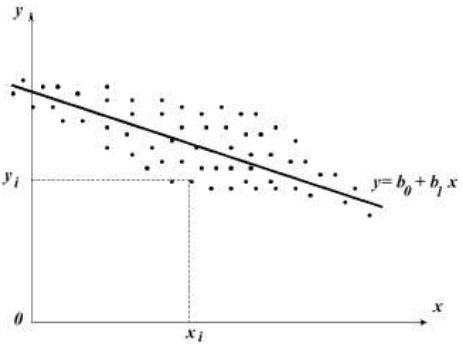
Якщо функція $f(x)$ має відповідні похідні, то система (1.3.5.13) визначається з (1.3.5.9). Якщо значення другої похідної рівні m_1=0, m_n =0,

то отримуємо нормальні сплайн - функції. У разі, коли $m_1 = m_n$ маємо періодичні сплайн - функції.

Рішення задачі інтерполяції функції зводиться до побудови інтерполяційного полінома Лагранжа, а у разі рівновіддалених вузлів – до побудови інтерполяційного полінома Ньютона.

При проведенні теоретичних і експериментальних досліджень часто виникає необхідність отримання більш простій емпіричної формули, яка добре відображає фізичні властивості досліджуваного процесу. Цього можна добитися з використанням апроксимації. Опис зв'язку між деяким числом N пара значень x_i і y_i із забезпеченням найменшої середньоквадратичної погрішності, можна здійснити з використанням методу найменших квадратів, шляхом апроксимації функцій ортогональними поліномами. З метою зменшення випадкових помилок і отримання біль гладкої функції згладжування часто використовують кубічні сплайни.

У справжній роботі ми зупинимося тільки на апроксимації функцій з використанням методу найменших квадратів. Аналіз методу почнемо з лінійної апроксимації. На рис. 1.3.5.1 показаний розподіл пар значень x_i, y_i і графік лінійної апроксимуючої залежності $y = y(x)$.



Рiс.1.3.5.1. Лінійна апроксимація

Геометрична суть процесу лінійної апроксимації полягає в тому, щоб провести пряму $y(x) = b_0 + b_1 x$ так, щоб величина всіх відхилень відповідала умові

$$\begin{aligned} U &= \sum_{i=1}^{N} U_i = min, \\ U_1 &= [y_1 - b_1 x_1 - b_0]^2, \\ U_2 &= [y_2 - b_1 x_2 - b_0]^2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \\ U_N &= [y_N - b_1 x_N - b_0]^2. \end{aligned} \qquad (1.3.5.14)$$

Для виконання першої умови системи (1.3.5.14) необхідно визначити відповідні приватні похідні

$$\frac{\partial U_i}{\partial b_0} = 0,\ \frac{\partial U_i}{\partial b_1} = 0,\ i = 1 \ldots N. \qquad (1.3.5.15)$$

Визначимо значення даних приватних похідних

$$\begin{gathered} \frac{\partial U_1}{\partial b_0} = -2[y_1 - b_1 x_1 - b_0],\ \frac{\partial U_1}{\partial b_1} = \\ = -2[y_1 - b_1 x_1 - b_0]x_1 = -2\left[y_1 x_1 - b_1 x_1^2 - b_0 x_1\right], \\ \frac{\partial U_2}{\partial b_0} = -2[y_2 - b_1 x_2 - b_0],\ \frac{\partial U_2}{\partial b_1} = \\ = -2[y_2 - b_1 x_2 - b_0]x_2 = -2\left[y_2 x_2 - b_1 x_2^2 - b_0 x_2\right], \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{\partial U_N}{\partial b_0} = -2[y_N - b_1 x_N - b_0],\ \frac{\partial U_N}{\partial b_1} = \\ = -2[y_N - b_1 x_N - b_0]x_N = -2\left[y_N x_N - b_1 x_N^2 - b_0 x_N\right]. \end{gathered} \qquad (1.3.5.16)$$

З урахуванням (1.3.5.15), (1.3.5.16) отримаємо

$$\sum_{i=1}^{N} \frac{\partial U_i}{\partial b_0} = 0,\ \sum_{i=1}^{N} \frac{\partial U_i}{\partial b_1} = 0. \qquad (1.3.5.17)$$

Вирішуючи спільно (1.3.5.16) і (1.3.5.17), отримаємо два рівняння для визначення постійних коефіцієнтів в лінійному рівнянні апроксимації

$$\begin{aligned}\sum_{i=1}^{N} y_i &= b_1 \sum_{i=1}^{N} x_i + N b_0,\\ \sum_{i=1}^{N} x_i y_i &= b_1 \sum_{i=1}^{N} x_i^2 + b_0 \sum_{i=1}^{N} x_i.\end{aligned} \qquad (1.3.5.18)$$

Вирішуючи систему двох лінійних рівнянь (1.3.5.18) щодо постійних коефіцієнтів лінійного рівняння апроксимації, отримаємо

$$\begin{aligned}b_0 &= \frac{1}{N}\left(\sum_{i=1}^{N} y_i - b_1 \sum_{i=1}^{N} x_i\right),\\ b_1 &= \frac{\sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i - N \sum_{i=1}^{N} x_i y_i}{\left(\sum_{i=1}^{N} x_i\right)^2 - N \sum_{i=1}^{N} x_i^2}.\end{aligned} \qquad (1.3.5.19)$$

Обчислення середньоквадратичної погрішності дозволить якісно оцінювати ступінь наближення крапок x_i, y_i до апроксимуючої прямої. Величина середньоквадратичної погрішності визначиться по формулі

$$\sigma^2 = \frac{1}{N}\left[\sum_{i=1}^{N} y_i^2 - b_1 \sum_{i=1}^{N} y_i - b_0 \sum_{i=1}^{N} x_i y_i\right].$$

Більш загальний випадок апроксимації пар значень x_i, y_i забезпечується за допомогою поліноміальної регресії вигляду

$$y(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m. \qquad (1.3.5.20)$$

Суть поліноміальної апроксимації полягає, з обліком (1.3.5.20), у виконанні наступної умови

$$\begin{aligned}U &= \sum_{i=1}^{N} U_i = \min,\\ U_1 &= \left[y_1 - a_0 - a_1 x_1 - a_2 x_1^2 - \ldots - a_m x_1^m\right]^2,\\ U_2 &= \left[y_2 - a_0 - a_1 x_2 - a_2 x_2^2 - \ldots - a_m x_2^m\right]^2,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ U_N &= \left[y_N - a_0 - a_1 x_N - a_2 x_N^2 - \ldots - a_m x_N^m\right]^2.\end{aligned} \qquad (1.3.5.21)$$

Для мінімізації величини U з обліком (1.3.5.21), визначимо значення приватних похідних по відповідних коефіцієнтах полінома (1.3.5.20)

$$\frac{\partial U_i}{\partial a_0} = \frac{\partial U_i}{\partial a_1} = \frac{\partial U_i}{\partial a_2} = ... = \frac{\partial U_i}{\partial a_m} = 0,\ \ i = 1...N,$$

$$\frac{\partial U_1}{\partial a_0} = -2\left[y_1 - a_0 - a_1 x_1 - a_2 x_1^2 - ... - a_m x_1^m\right],$$

$$\frac{\partial U_1}{\partial a_1} = -2\left[y_1 - a_0 - a_1 x_1 - a_2 x_1^2 - ... - a_m x_1^m\right] x_1,$$

$$\frac{\partial U_i}{\partial a_2} = -2\left[y_1 - a_0 - a_1 x_1 - a_2 x_1^2 - ... - a_m x_1^m\right] x_1^2,$$

$$\frac{\partial U_1}{\partial a_m} = -2\left[y_1 - a_0 - a_1 x_1 - a_2 x_1^2 - ... - a_m x_1^m\right] x_1^m, \qquad (1.3.5.22)$$

$$\frac{\partial U_N}{\partial a_0} = -2\left[y_N - a_0 - a_1 x_N - a_2 x_N^2 - ... - a_m x_N^m\right],$$

$$\frac{\partial U_N}{\partial a_1} = -2\left[y_N - a_0 - a_1 x_N - a_2 x_N^2 - ... - a_m x_N^m\right] x_N,$$

$$\frac{\partial U_N}{\partial a_2} = -2\left[y_N - a_0 - a_1 x_N - a_2 x_N^2 - ... - a_m x_N^m\right] x_N^2,$$

$$\frac{\partial U_N}{\partial a_m} = -2\left[y_N - a_0 - a_1 x_N - a_2 x_N^2 - ... - a_m x_N^m\right] x_N^m.$$

З урахуванням (1.3.5.21), отримаємо

$$\sum_{i=1}^{N} \frac{\partial U_i}{\partial a_0} = \sum_{i=1}^{N} \frac{\partial U_i}{\partial a_1} = \sum_{i=1}^{N} \frac{\partial U_i}{\partial a_2} = ... = \sum_{i=1}^{N} \frac{\partial U_i}{\partial a_m} = 0.$$

Використовуючи останню систему рівнянь можна визначити лінійну систему рівнянь для визначення постійних коефіцієнтів в поліноміальному апроксимуючому рівнянні (1.3.5.20)

$$\sum_{i=1}^{N} y_i = Na_0 + a_1 \sum_{i=1}^{N} x_i + a_2 \sum_{i=1}^{N} x_i^2 + ... + a_m \sum_{i=1}^{N} x_i^m,$$

$$\sum_{i=1}^{N} x_i y_i = a_0 \sum_{i=1}^{N} x_i + a_1 \sum_{i=1}^{N} x_i^2 + a_2 \sum_{i=1}^{N} x_i^3 + ... + a_m \sum_{i=1}^{N} x_i^{m+1},$$

$$\sum_{i=1}^{N} x_i^2 y_i = a_0 \sum_{i=1}^{N} x_i^2 + a_1 \sum_{i=1}^{N} x_i^3 + a_2 \sum_{i=1}^{N} x_i^4 + ... + a_m \sum_{i=1}^{N} x_i^{m+2}, \qquad (1.3.5.23)$$

$$\sum_{i=1}^{N} x_i^m y_i = a_0 \sum_{i=1}^{N} x_i^m + a_1 \sum_{i=1}^{N} x_i^{m+1} + a_2 \sum_{i=1}^{N} x_i^{m+2} + ... + a_m \sum_{i=1}^{N} x_i^{2m}.$$

Систему лінійних рівнянь (1.3.5.23) можна вирішити методом Гауса з вибором головного елементу (див. 1.3.1).

Поліноміальна регресія дозволяє апроксимувати N пара x_i, y_i з мінімальною середньоквадратичною погрішністю

$$E = \sqrt{\sum_{i=1}^{N} \frac{\varepsilon_i^2}{N+1}}. \qquad (1.3.5.24)$$

Поліноміальна апроксимація з автоматичним вибором ступеня полінома дозволяє шляхом підвищення ступеню полінома з $m = 1$ до $m = k$ визначати величини коефіцієнтів $a_0 ... a_m$.

По (1.3.5.24) визначається середньоквадратична погрішність E яка може порівнюватися із заданою E_1. При виконанні нерівності $E > E_1$ ступінь полінома m збільшується на 1, до того моменту, коли виконуватиметься нерівність $E < E_1$.

1.3.6. Математичні моделі, отримані шляхом обробки експериментальних даних

Розглянуті нижче методи планування експериментальних досліджень дозволяють отримувати математичні моделі з необхідною точністю і статистичною надійністю при мінімальних трудомісткості і матеріальних витратах.

Результатом будь-якого досвіду є подія, яка може мати якісну або кількісну характеристику. Події можуть бути достовірними, неможливими, випадковими. По характеру відповідності подій між собою їх можна класифікувати з одного боку як несумісні і сумісні, з іншого боку як залежні і незалежні.

У класичній схемі вірогідність $P(s)$ виникнення події S визначається відношенням числа елементарних випадків, сприятливих даній події SB до

загального числа всіх рівно можливих випадків SO

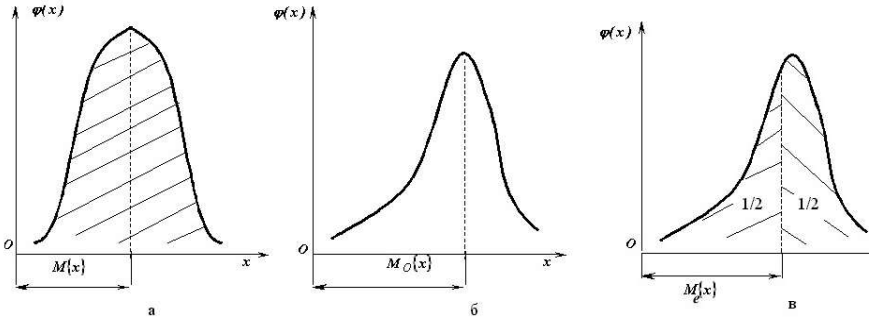
$$P(S) = \frac{SB}{SO}. \qquad (1.3.6.1)$$

Велике значення при проведенні експериментальних досліджень необхідно приділяти точність вимірювань. Отриманий експериментальним шляхом результат завжди містить деяку помилку. Розрізняють систематичні, випадкові і грубі помилки. Систематичні помилки з'являються унаслідок несправності вимірювальних приладів, недосконалості методики вимірювання. Випадкові помилки при повторних вимірюваннях можуть мати як об'єктивний, так і суб'єктивний характер (наприклад, короткочасна зміна температури навколишнього середовища, напруга живлення приладів і ін.). При проведенні вимірювань можуть виникати грубі помилки, які викликані неуважністю експериментатора. Для визначення випадкових і грубих помилок використовують методи математичної статистики, які засновані на елементах теорії вірогідності. Випадкові величини можуть бути дискретними і безперервними.

Безперервні випадкові величини характеризуються функцією розподілу $F(x)$ щільністю вірогідності $\varphi(x)$ які зв'язані співвідношенням [31]

$$F(x) = \int_{-\infty}^{x} \varphi(x)dx. \qquad (1.3.6.2)$$

На практиці широко використовуються числові характеристики (статистики) випадкових величин. До них відносяться математичне очікування $M\{x\} = a$ або середнє значення випадкової величини x (абсциса центру тяжіння площі під кривій щільності вірогідності $\varphi(x)$ - рис.1.3.6.1 а), мода випадкової величини $M_O\{x\}$ (абсциса відповідає максимальному значенню щільності вірогідності – рис.1.3.6.1 би)медіана випадкової величини $M_e\{x\}$ (абсциса точок прямої, яка ділить площу, обмежену кривій щільності вірогідності, навпіл – рис.1.3.6.1 в) [31].



Ріс.1.3.6.1. Графік щільності вірогідності з вказівкою: математичного очікування, моди і медіани

Дисперсія $D\{x\}$ (σ^2) є одній з основних характеристик розсіяння випадкової величини x біля центру розподілу і визначається по формулі

$$D\{x\} = \sigma^2 = \int_{-\infty}^{\infty} (x-a)^2 \varphi(x)dx. \qquad (1.3.6.3)$$

Середнім квадратичним відхиленням (стандартним відхиленням) називається позитивне значення квадратного кореня з дисперсії

$$\sigma = \sqrt{\sigma^2}. \qquad (1.3.6.4)$$

Відношення середнього квадратичного відхилення до математичного очікування називається коефіцієнтом варіації γ який визначається по формулі

$$\gamma = \frac{\sqrt{\int_{-\infty}^{\infty}\left[x - \int_{-\infty}^{\infty} x\varphi(x)dx\right]^2 \varphi(x)dx}}{\int_{-\infty}^{\infty} x\varphi(x)dx} = \frac{\sigma}{a}. \qquad (1.3.6.5)$$

Вираз (1.3.6.5) показує наскільки велике розсіяння в порівнянні з середнім значенням випадкової величини.

Показник S_k асиметрії розподілу визначається по формулі

$$S_k = \frac{\mu_3}{\sigma^3}, \quad \mu_3 = \int_{-\infty}^{\infty} (x-a)^3 \varphi(x)dx. \qquad (1.3.6.6)$$

Для оцінки гостро пікового розподілу (крутизні) використовується показник ексцесу E_k який визначається по формулі

$$E_k = \frac{\mu_4}{\sigma^4} - 3, \quad \mu_4 = \int_{-\infty}^{\infty} (x-a)^4 \varphi(x) dx. \qquad (1.3.6.7)$$

Визначені по (1.3.6.6).(1.3.6.7) показники асиметрії і ексцесу дорівнюють нулю у разі нормального розподілу.

Нижче приведені виразу для основних диференціальних функцій розподілу (щільності вірогідності) випадкових величин

нормальний розподіл Гауса

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-a)^2}{2\sigma^2}\right], \quad -\infty < x < \infty,$$

логарифмічний нормальний розподіл

$$\varphi(x) = \frac{1}{x\sigma_y\sqrt{2\pi}} \exp\left[-\frac{(\ln x - M\{\ln x\})^2}{2\sigma_y^2}\right], \quad \sigma_y^2 = D\{\ln x\}, \quad 0 \le x < \infty,$$

експоненціальне

$$\varphi(x) = \frac{1}{M\{x\}} \exp\left(-\frac{x}{M\{x\}}\right), \quad 0 \le x < \infty, \quad \frac{1}{M\{x\}} > 0,$$

показово-статечне

$$\varphi(x) = \frac{x^{M\{x\}-1}}{(M\{x\}-1)!} \exp(-x), \quad 0 \le x < \infty, \quad M\{x\} - 1 > 0,$$

Вейбулла - Гнеденко

$$\varphi(x) = \begin{cases} 1 - \exp\left[-\left(\frac{x - x_H}{c}\right)^b\right], & при \quad x > x_H, \\ 0, & при \quad x \le x_H, \end{cases}$$

$b > 0, c > 0, \quad x_H$ - нижня межа зміни випадкової величини

Бета - розподіл

$$\varphi(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

$$M\{x\} = \frac{\alpha}{\alpha+\beta}, \quad \sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}, \quad 0 \le x < \infty.$$

При проведенні експериментальних досліджень неможливо провести дуже багато виміри. З цього виходить, що не можна побудувати функцію розподілу для визначення дійсного значення x. Таким чином, хорошим наближенням до дійсного значення можна рахувати середньоарифметичне значення

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k, \qquad (1.3.6.8)$$

де n - число вимірювань.

Достатньо точною оцінкою помилки можна вважати вибіркову дисперсію S_n^2 витікаючи з відповідного (наприклад, нормального) закону розподілу, але що відноситься до кінцевого числа вимірювань. Така назва пояснюється тим, що зі всієї безлічі можливих значень x_k (генеральній сукупності) вимірюють лише кінцеве число значень n яке називається вибіркою і характеризується вибірковим середнім значенням і вибірковою дисперсією.

Середньою квадратичною погрішністю окремого вимірювання називається величина рівна кореню квадратному з вибіркової дисперсії

$$S_n = \sqrt{\frac{\sum_{k=1}^{n}(\bar{x} - x_k)^2}{n-1}}. \qquad (1.3.6.9)$$

Середня квадратична погрішність ряду вимірювань визначається по формулі

$$S_{\bar{x}} = \frac{S_n}{\sqrt{n}}. \qquad (1.3.6.10)$$

Аналіз залежності (1.3.6.10) показує, що із збільшенням числа вимірювань середня квадратична погрішність прагне до нуля. Проте, це досягається лише при вельми значному числі вимірювань.

Після визначення наближеного значення вимірюваної величини необхідно визначити надійність знайденого дійсного значення x. Довірчим інтервалом при цьому буде інтервал $(\bar{x}-\varepsilon;\bar{x}+\varepsilon)$ у якому знаходиться із заданою вірогідністю дійсне значення x ($\varepsilon = t \cdot S_{\bar{x}}$ $t = \frac{|\bar{x}-x|}{S_{\bar{x}}}$).

Якщо по n результатам розраховується середньоарифметичне значення $\bar{x}$ то квадрат експериментальної оцінки середньо квадратичного відхилення $S^2_{(x_k)}$ є експериментальною оцінкою дисперсії

$$S^2_{(x_k)} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})^2. \qquad (1.3.6.11)$$

Числом мір свободи f називається різниця між числом незалежних результатів в n повторах і числом рівнянь, в яких ці результати використані для розрахунку невідомих оцінок (в даному випадку вони використовувалися в одному рівнянні (1.3.6.8) для розрахунку середньоарифметичного значення $\bar{x}$).

Довірча помилка $\varepsilon_{(x_k)}$ для одиничного результату, з обліком (1.3.6.11), дорівнює $\varepsilon_{(x_k)} = t \cdot S_{(x_k)}$ а довірча помилка для середнього результату $\varepsilon_{(\bar{x})}$ з обліком (1.3.6.10), визначається по формулі $\varepsilon_{(\bar{x})} = t \cdot S_{(\bar{x})}$.

Відносну помилку середнього результату визначають по формулі

$$O_{(\bar{x})} = \frac{\varepsilon_{(\bar{x})}}{\bar{x}} 100\%. \qquad (1.3.6.12)$$

Відносне стандартне відхилення визначається як відношення стандартного відхилення до середньоарифметичного значення вимірюваної величини

$$\Delta = \frac{S_{(\bar{x})}}{\bar{x}} 100\%. \qquad (1.3.6.13)$$

Як практичне застосування розглянемо випадок визначення необхідної кількості повторних дослідів для отримання результатів із заданою точністю. На попередньому етапі визначуваний по (1.3.6.9).(1.3.6.10) величину середньої квадратичної погрішності $S_{(\bar{x})}$ для невеликого числа

дослідів (2-3 вимірювання). Задаємося довірчою вірогідністю α (для інженерних розрахунків ця величина дорівнює 0,95) і значенням довірчої помилки для середнього результату (зазвичай можна приймати $\varepsilon_{(\bar{x})} = (0{,}05...0{,}1)\cdot\bar{x}$). Тоді, з обліком (1.3.6.10).(1.3.6.11), отримаємо

$$t_p = \frac{\varepsilon_{(\bar{x})}}{S_{(x_k)}}\sqrt{n}. \qquad (1.3.6.14)$$

Для заданого значення довірчої вірогідності α і числа мір свободи $f = n-1$ визначаємо по таблиці 1.3.6.1 значення t- критерію Стьюдента t_T. По формулі (1.3.6.14) визначаємо розрахункове значення t_P- критерію Стьюдента. Для забезпечення точності результату із заданою довірчою помилкою $\varepsilon_{(\bar{x})}$ з довірчою вірогідністю α необхідно, щоб виконувалася нерівність $t_p > t_T$. Інакше, необхідно збільшити число повторів опиту n і по формулі (1.3.6.14) визначити нове значення розрахункового t_P- критерію Стьюдента. При визначенні нового табличного значення t_T- критерію Стьюдента необхідно враховувати зміну числа мір свободи $f = n-1$.

При обробці експериментальних даних, для визначення грубих помилок, можна використовувати t- критерій Стьюдента. Послідовність виконання операцій буде наступна. Для заданого числа повторів досвіду n визначаємо значення x_{kS} яке викликає сумнів. Відкидаємо його і для тих, що залишилися $n-1$ даних визначаємо середньоарифметичне значення $\bar{x}_{(n-1)}$ і середньо квадратичне відхилення (стандартне відхилення) $S_{(x_k)}$ по формулі

$$\bar{x}_{(n-1)} = \frac{\sum_{k=1}^{n-1} x_k}{n-1}, \quad S_{(x_k)} = \sqrt{\frac{\sum_{k=1}^{n-1}\left[\bar{x}_{(n-1)} - x_k\right]^2}{n-2}}.$$

Потім визначаємо розрахункове значення t_p- критерію Стьюдента по формулі

$$t_p = \frac{\left| x_{kS} - \bar{x}_{(n-1)} \right|}{S_{(x_k)}}. \qquad (1.3.6.15)$$

Результат досвіду вважається грубою помилкою, якщо розрахункове значення t_P по модулю більше табличного $t_p > t_T$. В даному випадку результат x_{kS} можна виключити як грубу помилку (табличне значення t_T - критерію Стьюдента вибирається для числа мір свободи $f = (n-1)-1$).

Сучасна математична теорія оптимального планування експерименту включає два основні напрями: вивчення механізмів складних процесів і поведінка багатокомпонентних систем на основі планування експерименту; оптимізація технологічних процесів і властивостей багатокомпонентних систем на основі планування експерименту. Експериментальні дослідження можуть проводитися безпосередньо на об'єкті або його моделі. Останнім часом разом з фізичними моделями всього більшого поширення набувають абстрактні математичні моделі. В цьому випадку говорять про проведення обчислювального експерименту.

Основними вимогами, що пред'являються до експерименту, є відтворюваність результатів експерименту і керованість об'єктами процесу.

Експериментальні дослідження можуть бути пасивними і активними. Якщо результати виходять при випадкових змінах вхідних параметрів, то говорять про пасивний експеримент. Якщо вхідні параметри змінюються по складеному експериментатором плану, то говорять про активний експеримент. Необхідно відзначити, що абсолютно керованих об'єктів дослідження не існує. Це пояснюється тим, що на об'єкт діють некеровані чинники, які впливають на відтворюваність результатів експерименту. Якщо всі чинники, що діють, є некерованими, то основним інструментом для дослідження поведінки об'єкту є пасивний експеримент.

Нижче буде розглянута тільки методика проведення і обробки отриманих даних для активних експериментів.

При проведенні експериментальних досліджень чинники можуть бути представлені в натуральних і кодованих величинах. Для виконання кодування використовують формулу

$$x_i = \frac{X_i - X_{iO}}{h_i}, \qquad (1.3.6.16)$$

де x_i - значення чинника в кодованих змінних (може приймати значення – 1, 0 +1); X_i - значення чинника на одному з рівнів в натуральних величинах; X_{iO} - значення чинника на нульовому (основному) рівні в натуральних величинах; h_i - інтервал варіювання; i - номер чинника.

При плануванні експерименту основна увага повинна приділятися вибору оптимального плану, коли при мінімальному числі дослідів дослідник отримує максимальну інформацію про об'єкт. Так для лінійних моделей, при числі чинників $k \le 5$ вельми ефективним є повний факторний експеримент (ПФЕ) або дробовий факторний експеримент (ДФЕ). Дані плани володіють ортогональністю, ротатабельністю (рух від центру експерименту на всіх напрямках є рівнозначним) і симетричністю щодо центру експерименту.

Для моделей другого порядку широко використовуються ортогональні, ротатабельні (Боксу) D - оптимальні плани (Кифера). Вельми характерним є число дослідів N у плані. Так для ортогонального плану для $k=2$ $N=9$, для $k=3$ $N=15$, для $k=4$ $N=25$. Для ротатабельного плану (Боксу) при $k=2$ $N=13$, при $k=3$ $N=20$, при $k=4$ $N=31$. Для D - оптимального плану (Кифера) при $k=2$ $N=9$, при $k=3$ $N=26$, при $k=4$ $N=72$.

Зупинимося на розгляді ПФЕ (у нім реалізуються всі можливі комбінації даних рівнів чинників). Загальне число дослідів визначається з виразу

$$N = 2^k. \qquad (1.3.6.17)$$

Графічна інтерпретація ПФЕ для двох чинників показана на рис.1.3.6.2 а. Матриця ПФЕ включає: стовпці, відповідні кожному чиннику; стовпці, відповідні взаємодіям чинників. Число рядків визначається по (1.3.6.17). У

таблиці 1.3.6.2 показана матриця ПФЕ для двох чинників.

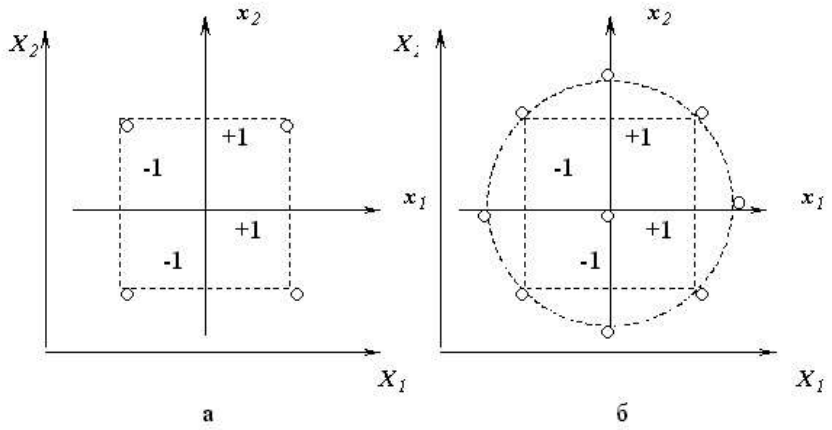


Рис.1.3.6.2. Графічна інтерпретація ПФЕ і ротатабельного плану для 2 чинників

Таблиця 1.3.6.2. Матриця ПФЕ для двох чинників

Номер досвіду	Чинники		
	x_1	x_2	x_1x_2
1	+	+	+
2	-	+	-
3	+	-	-
4	-	-	+

Рівняння регресії для двох і трьох чинників мають вигляд [32]

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_{12}x_1x_2,$$
$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 + b_{123}x_1x_2x_3. \quad (1.3.6.18)$$

Коефіцієнти в рівняннях регресії (1.3.6.18) визначаються по наступних формулах

$$b_0 = \frac{\sum_1^N \bar{y}_u}{N}, \quad b_i = \frac{\sum_1^N x_{iu}\bar{y}_u}{N}, \quad b_{ij} = \frac{\sum_1^N x_{iu}x_{ju}\bar{y}_u}{N}, \quad b_{ijk} = \frac{\sum_1^N x_{iu}x_{ju}x_{ku}\bar{y}_u}{N}, \quad (1.3.6.19)$$

де $\bar{y}_u$ - середньоарифметичне значення критерію оптимізації для кожного досвіду в матриці (залежить від числа повторних вимірювань).

При збільшенні числа чинників $k > 5$ проведення експерименту за планом ПФЕ стає вельми трудомістким. В цьому випадку використовують

тільки певну частину ПФЕ. Такий підхід називається проведенням експерименту по матриці ДФЕ. Дробові репліки підрозділяються на регулярних і нерегулярних. Перші виходять з матриці ПФЕ діленням її на число частин, кратне 2 в якому-небудь ступені. При постановці ДФЕ виходять з припущення, що коефіцієнти регресії при взаємодіях вищих порядків (потрійних і вище) незначуще відрізняються від нуля. Тоді стає можливим використовувати в плані ПФЕ стовпці, що відносяться до взаємодій вищих порядків, для оцінки додаткових лінійних ефектів.

Оцінка значущості коефіцієнтів регресії (1.3.6.19) базується на визначенні довірчого інтервалу (коефіцієнт буде значущим якщо його абсолютна величина більше довірчого інтервалу), який рівний $2\Delta b_i$. Величина Δb_i визначається по формулі

$$\Delta b_i = \frac{t \cdot S_{\{y\}}}{\sqrt{N_1 n}}, \quad S_{\{y\}} = \sqrt{\frac{\sum_{1}^{N}\sum_{1}^{n}(y_{uj} - \bar{y}_u)^2}{N(n-1)}}, \qquad (1.3.6.20)$$

де n - число спостережень в окремому досвіді; N - число рядків матриці планування експерименту; t - критерій Стьюдента (див. таблицю 1.3.6.1); N_1 - число дослідів, що враховуються при розрахунку коефіцієнта.

Ротатабільне планування другого порядку використовується у разі, коли поверхня відгуку носить явно нелінійний характер. В зв'язку з цим варіювання чинників необхідно здійснювати на трьох і п'яти рівнях. Ротатабельний план виходить з плану ПФЕ шляхом додавання до нього певної кількості «зоряних» і нульових крапок. «Зоряні» крапки розташовуються на осях на відстані $2^{\frac{k}{4}}$ від нульової крапки, яке називається плечем. На рис.1.3.6.2 би показана графічна інтерпретація ротатабельного плану другого порядку для двох чинників. У таблиці 1.3.6.3 представлені основні співвідношення між числом чинників і загальним числом дослідів для ротатабельного плану другого порядку.

Таблиця 1.3.6.3. Залежність числа дослідів від числа чинників для

ротатабельного плану другого порядку

Число чинників k	Число точок «ядра» $n_я$	Число «зоряних» крапок	Число нульових крапок	Величина плеча для «зоряних» крапок	Загальне число дослідів N
2	4	4	5	1,414	13
3	8	6	6	1,682	20
4	16	8	7	2,000	31
5	32	10	10	2,378	52

У таблиці 1.3.6.4 представлена матриця планування експерименту для ротатабельного плану другого порядку для 2 чинників

Таблиця 1.3.6.4. Матриця планування експерименту

Номер досвіду	Чинники		
	x_1	x_2	x_1x_2
1	+	+	+
2	-	+	-
3	+	-	-
4	-	-	+
5	-1,414	0	0
6	+1,414	0	0
7	0	-1,414	0
8	0	+1,414	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0

Коефіцієнти в рівнянні регресії для ротатабельного плану визначаються по наступних формулах [32]

$$b_0 = a_1\sum_{j=1}^{N} y_j - a_2\sum_{i=1}^{k}\sum_{j=1}^{N} x_{ij}^2 y_j,\quad b_i = a_3\sum_{j=1}^{N} x_{ij} y_j,$$
$$b_{ij} = a_4\sum_{j=1}^{N} x_{ij}x_{jj} y_j,\quad b_{ii} = a_5\sum_{j=1}^{N} x_{ij}^2 y_j + a_6\sum_{i=1}^{k}\sum_{j=1}^{N} x_{ij}^2 y_j - a_7\sum_{j=1}^{N} y_j, \qquad (1.3.6.21)$$

де $a_1 \ldots a_7$ - постійні коефіцієнти, визначувані з таблиці 1.3.6.5.

Таблиця 1.3.6.5. Значення коефіцієнтів $a_1 \ldots a_7$ у виразі (1.3.6.21)

Число чинників k	Число дослідів N	Коефіцієнти						
		a_1	a_2	a_3	a_4	a_5	a_6	a_7
2	13	0,2	0,1	0,125	0,25	0,125	0,0187	0,1
3	20	0,1663	0,0568	0,0732	0,125	0,0625	0,0069	0,0568
4	31	0,1428	0,0357	0,0417	0,0625	0,0312	0,0037	0,0357
5	52	0,0988	0,0191	0,0231	0,0312	0,0156	0,0015	0,0191

Величина довірчого інтервалу для коефіцієнтів в рівнянні регресії при ротатабельном плануванні другого порядку визначається по формулах

$$\Delta b_0 = \pm 2S_{\{b_0\}},\quad \Delta b_i = \pm 2S_{\{b_i\}}, \Delta b_{ii} = \pm 2S_{\{b_{ii}\}},\quad \Delta b_{ij} = \pm 2S_{\{b_{ij}\}},$$
$$S_{\{b_0\}} = \sqrt{a_8}\,S_{\{\bar{y}\}},\quad S_{\{b_i\}} = \sqrt{a_9}\,S_{\{\bar{y}\}},\quad S_{\{b_{ii}\}} = \sqrt{a_{10}}\,S_{\{\bar{y}\}},\quad S_{\{b_{ij}\}} = \sqrt{a_{11}}\,S_{\{\bar{y}\}}, \qquad (1.3.6.22)$$
$$S_{\{\bar{y}\}} = \frac{S_{\{y\}}}{\sqrt{n}},\quad S_{\{y\}} = \sqrt{\frac{\sum_{1}^{N}\sum_{1}^{n}(y_{uj} - \bar{y}_u)^2}{N(n-1)}}.$$

Значення коефіцієнтів $a_8 \ldots a_{11}$ приведені в таблиці 1.3.6.6 [32].

Таблиця 1.3.6.6. Значення коефіцієнтів $a_8 \ldots a_{11}$ у виразі (1.3.6.22)

Число чинників k	Число дослідів N	Коефіцієнти			
		a_8	a_9	a_{10}	a_{11}
2	13	0,2	0,125	0,1438	0,25
3	20	0,1663	0,0732	0,0694	0,125
4	31	0,1428	0,0417	0,0341	0,0625

При ортогональному плануванні другого порядку матрицю планування ПФЕ добудовують шляхом введення нульової крапки і певного числа «зоряних» крапок. У таблиці 1.3.6.7 приведені основні характеристики

ортогонального плану для різного числа чинників.

Таблиця 1.3.6.7. Ортогональні плани другого порядку

Число чинників k	Число точок «ядра» $n_я$	Число «зоряних» крапок	Число нульових крапок	Величина плеча для «зоряних» крапок	Загальне число дослідів N
2	4	4	1	1,000	9
3	8	6	1	1,215	15
4	16	8	1	1,414	25

Коефіцієнти в рівнянні регресії для ортогонального планування другого порядку визначаються по формулах

$$b_i = \frac{\sum_{1}^{N_1} x_{iu} y_u}{N_1}, \quad b_{ij} = \frac{\sum_{1}^{N_2} x_{iu} x_{ju} y_u}{N_2}, \quad b_{ii} = \frac{\sum_{1}^{N} (x'_{iu})^2 y_u}{\sum_{1}^{N} (x'_{iu})^2},$$
$$b_0 = \frac{\sum_{1}^{N} y_u}{N} - \frac{\sum_{1}^{N} x_{iu}^2}{N} \sum_{1}^{k} b_{ii}, \quad (x'_{iu})^2 = x_{iu}^2 - \frac{\sum_{1}^{N} x_{iu}^2}{N}, \tag{1.3.6.23}$$

де N_1 - число дослідів не в нульових крапках; N_2 - число відмінних від нуля парних творів чинників.

При ортогональному плануванні для визначення довірчого інтервалу $2\Delta b_i$ користуються наступною залежністю

$$\Delta b_i = \pm 2 S_{\{b_i\}}, \quad S_{\{b_i\}} = \frac{S_{\{\bar{y}\}}}{\sqrt{A}}, \quad S_{\{\bar{y}\}} = \frac{S_{\{y\}}}{\sqrt{n}}, \quad S_{\{y\}} = \sqrt{\frac{\sum_{1}^{N} \sum_{1}^{n} (y_{uj} - \bar{y}_u)^2}{N(n-1)}}. \tag{1.3.6.24}$$

Значення коефіцієнта A визначаємо з таблиці 1.3.6.8 [32].

Таблиця 1.3.6.8. Значення коефіцієнта A у виразі (1.3.6.24)

Число	A

чинників k	b_0	b_i	b_{ij}	b_{ii}
2	9	6	4	2
3	15	10,94	8	4,34
4	25	20	16	8

Перевірка отриманих рівнянь регресії здійснюється за допомогою критерію Фішера. Розрахункове значення критерію Фішера визначається по формулі

$$F_p = \frac{S^2_{ad}}{S^2_{\{y\}}}, \qquad (1.3.6.25)$$

де S^2_{ad} - дисперсія адекватності рівняння регресії; $S^2_{\{y\}}$ - дисперсія відтворюваності.

Для ПФЕ

$$S^2_{ad} = \frac{\sum_1^N n(\bar{y}_u - \hat{y}_u)^2}{N-k-1}, \quad S^2_{\{y\}} = \frac{\sum_1^N \sum_1^n (y_{uj} - \bar{y}_u)^2}{N(n-1)}.$$

Для ортогонального планування другого порядку

$$S^2_{ad} = \frac{\sum_1^N n(\bar{y}_u - \hat{y}_u)^2}{N - \frac{(k+2)(k+1)}{2}}, \quad S^2_{\{y\}} = \frac{\sum_1^N \sum_1^n (y_{uj} - \bar{y}_u)^2}{N(n-1)}.$$

Для ротатабельного планування другого порядку

$$S^2_{ad} = \frac{\sum_1^N n(\bar{y}_u - \hat{y}_u)^2}{N - \frac{(k+2)(k+1)}{2} - (n_0 - 1)}, \quad S^2_{\{y\}} = \frac{\sum_1^N \sum_1^n (y_{uj} - \bar{y}_u)^2}{N(n-1)}.$$

де n_0 - число повторів в центрі експерименту.

На наступному етапі, після визначення по (1.3.6.25) розрахункового значення критерію Фішера, визначаємо табличне значення критерію Фішера F_T.

Число мір свободи для більшої дисперсії визначається по формулах: для ПФЕ

$$f_{a\partial} = N - k - 1,$$

для ортогонального планування другого порядку

$$f_{a\partial} = N - \frac{(k+2)(k+1)}{2},$$

для ротатабельного планування другого порядку

$$f_{a\partial} = N - \frac{(k+2)(k+1)}{2} - (n_0 - 1).$$

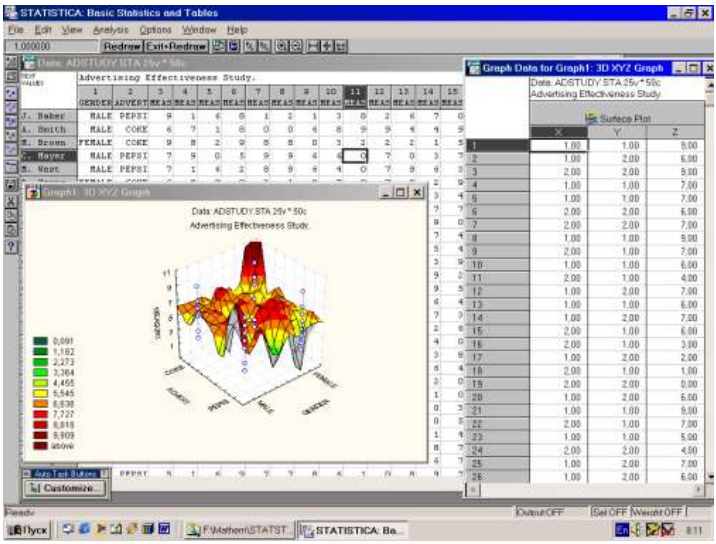
Число мір свободи для меншої дисперсії визначається по формулі

$$f_e = N(n-1).$$

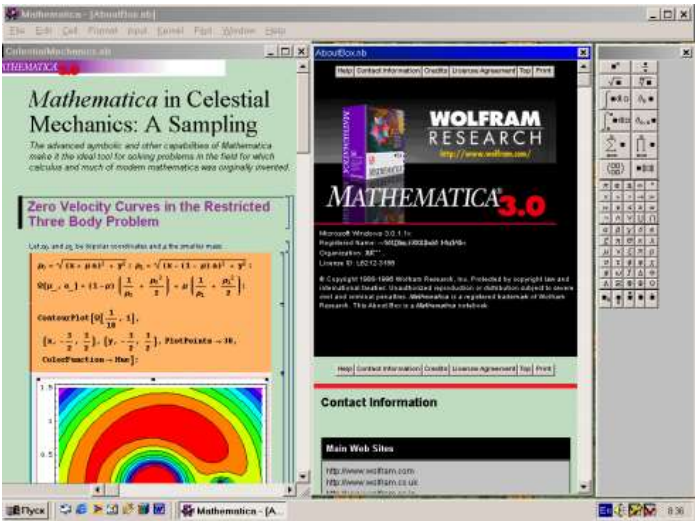
Для того, щоб регресійна модель, при вибраній довірчій вірогідності, адекватно описувала досліджуваний процес необхідне виконання умови

$$F_p < F_T. \qquad (1.3.6.26)$$

З розвитком інформаційних технологій велике значення придбали різні прикладні програмні пакети для статистичної обробки даних і реалізації активного планування експерименту. На рис.1.3.6.3а,б показані робочі вікна пакетів ***STATISTICA 5.0*** і ***Mathematica3***. На рис.1.3.6.4а,б показані робочі вікна пакетів ***SPSS*** і ***STATGRAPHI CS***.

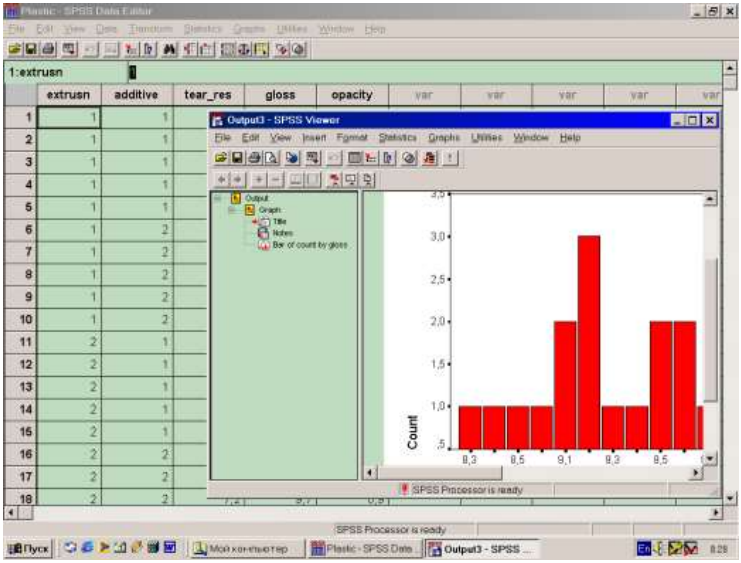


а

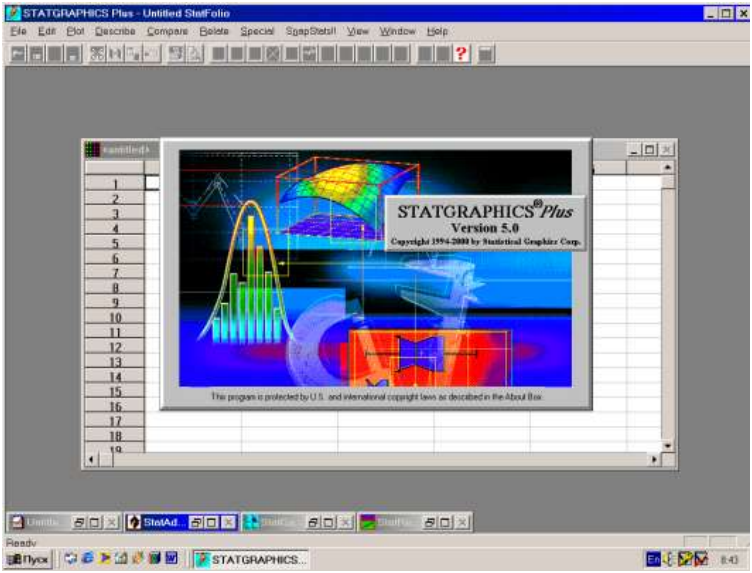


б

Рис.1.3.6.3. Робочі вікна пакетів ***STATISTICA 5.0*** і ***Mathematica3***



а



б

Рис.1.3.6.4. Робочі вікна пакетів ***SPSS*** і ***STATGRAPHI CS***

Статистичний пакет ***STATISTICA*** (проводиться фірмою Stat Soft Inc. США) включає наступні статистичні модулі: непараметрична статистика, дисперсійний аналіз, множинна регресія, нелінійне оцінювання, аналіз тимчасових рядів, кластерний і факторний аналіз, функціональний аналіз дискримінанта, канонічна кореляція, моделювання структурних рівнянь, аналіз процесів, планування експерименту і ін.

У 1995 році вийшла одна з останніх версій даного пакету ***STATISTICA 5.0*** для Windows, в якій статистичний аналіз експериментальних даних розбитий на наступні етапи: введення даних; використання певної процедури статистичної обробки даних; виведення результатів на дисплей або друк.

Статистична обробка даних в пакеті символьної математики ***Mathematica3*** міститься в пакеті розширення системи ***STATISTI*** який

орієнтований на статистичну обробку даних. Даний пакет містить наступні функції: ***Mean***- повертає середнє значення даних $\{x_1, x_2, ...x_n\}$; ***Median***- повертає медіану даних $\{x_1, x_2, ...x_n\}$; ***StandardDeviation*** - повертає стандартне середньо квадратичне відхилення для даних $\{x_1, x_2, ...x_n\}$; ***Variance***- повертає варіацію для даних $\{x_1, x_2, ...x_n\}$.

Апроксимація і інтерполяція експериментальних даних здійснюється з використанням наступних функцій: ***InterpolatingFunction*** - повертає функцію, значення якої у вузлових точках визначаються за допомогою інтерполяції; ***InterpolatingPolynomial*** - повертає інтерполяційний поліном; ***Interpolation*** - повертає функцію, що наближає; ***InterpolationOrder*** - повертає функцію (статечною поліном), що наближає, з вказаним ступенем.

Для вирішення завдань лінійної і нелінійної регресії використовується функція ***Fit***. Отримані результати забезпечують мінімальну середньо квадратичну погрішність між значеннями регресійної залежності і експериментальними даними для певних значень варійованого параметра.

Серед прикладних пакетів для обробки експериментальних даних можна виділити ***SPSS*** (розробник SPSS Inc.) і ***STATGRAPHICS*** (STATistical Graphics System) – розробник Manugistic Inc. [35]. Останній пакет ***STATGRAPHICS Plus for Windows***. Даний пакет має модульну структуру: ***Describe***- містить статистичні методи аналізу для одно- і багатьох змінних; ***Compare*** - містить процедури одно- і багатофакторного дисперсійного аналізу, методи порівняння декількох вибірок даних; ***Relate*** - включає операції простого поліноміального і множинного регресійного аналізу; ***Special*** - включає контроль якості, планування експерименту, аналіз тимчасових рядів, багатовимірні методи, розширений регресійний аналіз. ***Design of Experiment*** (Планування експерименту) включає повні і дробові плани, плани Плакетта-Бурмена, блокові плани, центральні композиційні плани, трирівневі факторіали, плани Бокса-Бенкена, плани Дрейпера-Лана,

прості центроідні і латинські плани.

Описані вище пакети прикладних програм для статистичної обробки експериментальних даних і активного планування експерименту дозволяють значно скоротити час отримання математичної моделі.

У справжній роботі нами для аналізу геометричної форми поверхні відгуку використовувалися інваріанти поверхонь другого порядку [1]. Їх використання дозволяє досягти поставленої мети без приведення рівнянь до канонічного вигляду.

Суть методу полягає в тому, що для безлічі точок тривимірного простору, координати яких в координатній системі задовольняють рівнянню алгебри 2-го ступеня

$$a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz + 2a_{14}x + 2a_{24}y + \\ + 2a_{34}z + a_{44} = 0, \quad (1.3.6.42)$$

визначають основні інваріанти ***S, T, ф, U*** і інваріанти ***Uг, UU***. Потім по таблиці 1.3.6.15 визначають, який геометричний образ має поверхню відгуку для тієї або іншої регресійної залежності (1.3.6.27.1.3.6.41).

Основні інваріанти і інваріанти визначаються як визначники і їх суми відповідних матриць коефіцієнтів многочлена алгебри. Формули для розрахунку приведені у вигляді системи (1.3.6.43). Перетворимо регресійні залежності (1.3.6.27).(1.3.6.41) до вигляду (1.3.6.42), для чого підставимо в останні значення кута обхвату і швидкості руху нитки, які відповідатимуть своїм мінімальним значенням (див.

$$S = a_{11} + a_{22} + a_{33};$$

$$\delta = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{vmatrix};$$

$$T = \begin{vmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{13} \\ a_{13} & a_{33} \end{vmatrix} + \begin{vmatrix} a_{22} & a_{23} \\ a_{23} & a_{33} \end{vmatrix};$$

$$\Delta = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{vmatrix}; \qquad (1.3.6.43)$$

$$\Delta' = \begin{vmatrix} a_{22} & a_{23} & a_{24} \\ a_{23} & a_{33} & a_{34} \\ a_{24} & a_{34} & a_{44} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{13} & a_{14} \\ a_{13} & a_{33} & a_{34} \\ a_{14} & a_{34} & a_{44} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{12} & a_{14} \\ a_{12} & a_{22} & a_{24} \\ a_{14} & a_{24} & a_{44} \end{vmatrix};$$

$$\Delta'' = \begin{vmatrix} a_{11} & a_{14} \\ a_{14} & a_{44} \end{vmatrix} + \begin{vmatrix} a_{22} & a_{24} \\ a_{24} & a_{44} \end{vmatrix} + \begin{vmatrix} a_{33} & a_{34} \\ a_{34} & a_{44} \end{vmatrix}.$$

план проведення експерименту табл. 1.3.6.11).

При підстановці ***ц=2,44*** радий швидкість ***x=0,9*** м/с. Такі значення даних величин вибрані нами виходячи з умови найменшої величини натягнення ведучої галузь нитки при варіюванні цих чинників.

Виконуючи перетворення залежностей (1.3.6.27).(1.3.6.41) до (1.3.6.42), вважали, що ***x=R, y=P0, z=P***. Відповідним чином були розподілені і коефіцієнти при конкретних величинах.

Результати розрахунку по формулах (1.3.6.43) інваріантів і інваріантів також приведені в таблиці 1.3.6.15.

Аналіз отриманих результатів дозволив встановити, що всі геометричні образи поверхонь відгуку зводяться до еліптичного параболоїда, параболічного циліндра і пари паралельних площин (див. рис. 1.3.6.9). З цього можна зробити вивід - при перетині поверхонь відгуку площинами у нас виходитимуть параболи і прямі лінії. Ця обставина враховувалася нижче при дослідженні впливу радіусу кривизни такою, що направляє, швидкості руху нитки, натягнення веденої гілки на

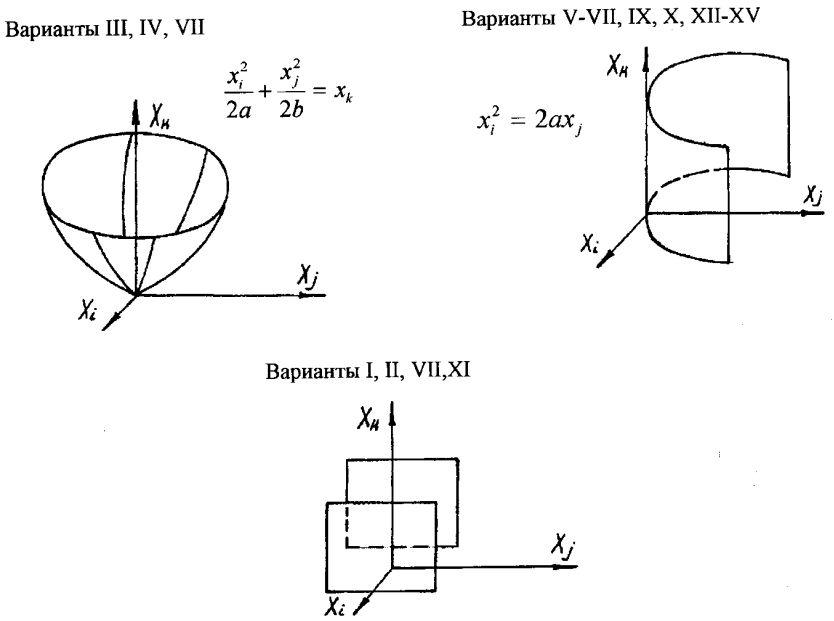


Рис.1.3.6.9. Геометричні образи поверхонь відгуку

натягнення ведучої гілки.

Такий диференційований підхід до вивчення регресійних залежностей (1.3.6.27) -(1.3.6.41) необхідний, оскільки велике число чинників утрудняє процес аналізу.

1.3.7. Питання оптимізації

Найважливішою умовою науково-технічного прогресу є пошук якнайкращого варіанту, з погляду наміченої мети, при проектуванні складних механічних систем, економічному плануванні, розподілі обмежених ресурсів, вдосконаленні технологічних процесів. Відшукання найменшого (найбільшого) значення цільової функції є основним

завданням оптимізації. Властивості цільової функції впливають на постановку завдання оптимізації і методи її дослідження. Завдання оптимізації можна трактувати як екстремальні завдання, методи вирішення яких розглядаються в теорії екстремальних завдань.

У багатьох завданнях економічного змісту зустрічаються опуклі і лінійні функції і множини. У 30-і роки радянським математиком Канторовічем Л.В. були розроблені нові напрями в теорії екстремальних завдань – лінійне і опукле програмування.

Теорія оптимального управління, доповнюючи розділ варіаційного числення, дозволяє вирішувати завдання управління технологічними процесами, приладами, системами.

Питаннями, присвяченими теорії і методам знаходження екстремумів функцій багатьох змінних, займається математичне програмування. При цьому, необхідно мати додаткові обмеження на ці змінні, які мають форму рівності і (або) нерівностей [36]. Залежно від властивостей цільової функції і додаткових обмежень, що накладаються на дані змінні, математичне програмування розбивається на лінійне і нелінійне програмування, дискретне, параметричне і стохастичне програмування. У разі, коли пошук оптимального рішення (залежно від виду цільової функції і обмежень) здійснюється у вигляді багатокрокового процесу, то говорять про динамічне програмування.

Зупинимося на загальних питаннях постановки і методів вирішення одновимірних завдань оптимізації. Математична формалізація завдання, в даному випадку, може бути записана таким чином: визначити найменше (найбільше) значення цільової функції $f(x)$ і значення змінної x за умови коли $x \in X$ (X - деяка множина).

Для цього скористаємося теоремами Вейерштраса і Ферма [14]. Теорема Вейерштраса свідчить, що всяка цільова функція $f(x)$

безперервна на відрізку $[a,b]$ приймає на цьому відрізку своє найменше $f(x_1)$ і найбільше $f(x_2)$ значення ($x_1 \in [a,b], x_2 \in [a,b]$)

$$f(x_1) \le f(x) \le f(x_2). \qquad (4.7.1)$$

У загальному випадку, цільова функція на заданому відрізку може досягати найменшого (найбільшого) значення в декількох крапках (наприклад, періодичні функції). Хай цільова функція $f(x)$ визначена на відрізку $[a,b]$. Можна стверджувати, що крапка x_1 (або x_2) доставляє локальний мінімум (максимум) якщо можна вказати таке ε що для всіх крапок x з відрізання $[a,b]$ для яких $|x-x_1|<\varepsilon$ (або $|x-x_2|<\varepsilon$), виконується нерівність $f(x) \ge f(x_1)$ (або $f(x) \le f(x_2)$).

Для випадку, коли цільова функція безперервна не на відрізку, а на всій осі, то можна скористатися наслідком з теореми Вейерштраса: якщо цільова функція $f(x)$ безперервна на всій прямій і якщо $\lim\limits_{x\to\infty} f(x) = \lim\limits_{x\to-\infty} f(x) = \infty$ те рішення задачі пошуку мінімуму (максимуму) цільової функції існує. Дане слідство можна використовувати і у разі, коли цільова функція безперервна на промені вигляду $a \le x < \infty$ або $a < x < \infty$.

Теорема Ферма [14] формулюється таким чином: якщо цільова функція диференцюєма в точці x_1 (для максимуму x_2) і якщо цільова функція має локальний екстремум в цій крапці, то $f'(x_1)=0$ (або $f'(x_2)=0$).

Крапки, для яких $f'(x)=0$ називаються стаціонарними. Стаціонарні крапки спільно з кінцевими крапками називаються критичними.

Співвідношення $f'(x_1)=0$ (або $f'(x_2)=0$) є лише необхідною умовою екстремуму. Спираючись на теореми Вейерштраса і Ферма можна стверджувати, що якщо відрізок $[a,b]$ кінцевий, цільова функція $f(x)$ безперервна на відрізку $[a,b]$ і диференцюєма у внутрішніх крапках $a < x < b$ те рішення знаходиться серед критичних крапок.

Вирішуючи рівняння

$$f'(x)=0, \qquad (4.7.2)$$

визначаємо дійсне його коріння, а також ті крапки, в яких похідна не існує. Розміщуємо всі критичні крапки в порядку зростання їх абсцис. У кожному з отриманих інтервалів вибирають довільну крапку і встановлюють в цій крапці знак першої похідної. Розглядають знаки $f'(x)$ у двох сусідніх інтервалах, послідовно переходячи зліва направо від першого інтервалу до останнього. Якщо похідна $f'(x)$ міняє знак з «-» на «+», то функція має мінімум в даній критичній крапці. Якщо знаки змінюються з «+» на «-», то функція має в критичній крапці максимум. Якщо ж в двох сусідніх інтервалах похідна $f'(x)$ не міняє знаку, то екстремуму в даній критичній крапці немає.

Приведені вище міркування безпосередньо пов'язані з поняттям опуклої функції. Функція називається опуклою, якщо для будь-якої хорди, що сполучає дві точки графіка цієї функції, її графік в проміжних крапках лежить нижче за цю хорду.

Перейдемо до дослідження на екстремум функції $f(x_1,x_2,...x_n)$ декілька змінних $x_1,x_2,...x_n$ вважаючи, що вони незалежні між собою. Необхідною умовою існування екстремуму функції $f(x_1,x_2,...x_n)$ є наступне: якщо функція $f(x_1,x_2,...x_n)$ досягає екстремуму при $x_1=x_{10},x_2=x_{20},...x_n=x_{n0}$ то кожна приватна похідна першого порядку функції $f(x_1,x_2,...x_n)$ по відповідній незалежній змінній $x_1,x_2,...x_n$ звертається в нуль при цих значеннях аргументів або не існує [37]

$$\left[\frac{\partial f(x_1,x_2,...x_n)}{\partial x_1}\right]_{\substack{x_1=x_{10}\\x_2=x_{20}\\.....\\x_n=x_{n0}}}=0,$$

$$\left[\frac{\partial f(x_1,x_2,...x_n)}{\partial x_2}\right]_{\substack{x_1=x_{10}\\x_2=x_{20}\\.....\\x_n=x_{n0}}}=0, \qquad (4.7.3)$$

$$..... ....$$

$$\left[\frac{\partial f(x_1,x_2,...x_n)}{\partial x_n}\right]_{\substack{x_1=x_{10}\\x_2=x_{20}\\.....\\x_n=x_{n0}}}=0.$$

Значення $x_{10},x_{20},...x_{n0}$ називаються критичними точками функції $f(x_1,x_2,...x_n)$. При дослідженні значення функції в критичних крапках можна підійти до визначення достатніх умов існування екстремуму. Для виведення достатніх умов використовують формулу Тейлора для представлення функції у вигляді суми многочлена ступеня n ($n=0,1,2,..$) і залишкового члена [14]. Хай в деякій області, що містить крапку $M_0(x_{10},x_{20},...x_{n0})$ функція $f(x_1,x_2,...x_n)$ має безперервні приватні похідні до третього порядку включно. Хай крапка $M_0(x_{10},x_{20},...x_{n0})$ є критичною крапкою для функції $f(x_1,x_2,...x_n)$ тоді

$$\left[\frac{\partial f(x_{10},x_{20},...x_{n0})}{\partial x_1}\right]=0,$$

$$\left[\frac{\partial f(x_{10},x_{20},...x_{n0})}{\partial x_2}\right]=0, \qquad (4.7.4)$$

$$..... ....$$

$$\left[\frac{\partial f(x_{10},x_{20},...x_{n0})}{\partial x_n}\right]=0.$$

Достатні умови існування екстремуму, наприклад для функції $f(x_1,x_2)$ два змінних x_1,x_2 мають вигляд [37]:

функція $f(x_1,x_2)$ має максимум, якщо

$$\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2 f(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2>0,\quad \frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}<0,$$

функція $f(x_1,x_2)$ має мінімум, якщо

$$\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2 f(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2>0,\quad \frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}>0,$$

функція $f(x_1,x_2)$ не має ні максимуму, ні мінімуму, якщо

$$\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2 f(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2<0,$$

функція $f(x_1,x_2)$ може мати екстремум в крапці $M_0(x_{10},x_{20})$ а може і не мати, якщо

$$\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2 f(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2 f(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2=0.$$

У тих випадках, коли змінні $x_1,x_2,...x_n$ є незалежними можна говорити про відшукання абсолютних мінімумів і максимумів. У разі, коли змінні зв'язані один з одним деякими додатковими умовами, говорять про умовні (відносних) екстремуми [14,37]. У загальному випадку змінні $x_1,x_2,...x_n$ зв'язані m ($m<n$) рівняннями

$$\begin{aligned}&\varphi_1(x_1,x_2,...x_n)=0,\\&\varphi_2(x_1,x_2,...x_n)=0,\\&\quad\cdots\cdots\cdots\cdots\\&\varphi_m(x_1,x_2,...x_n)=0.\end{aligned}\qquad(4.7.5)$$

Необхідні умови існування екстремуму виходять шляхом визначення приватних похідних від функції

$$\begin{aligned}\Phi(x_1,x_2,...x_n,\lambda_1,...\lambda_n)&=f(x_1,x_2,...x_n)+\lambda_1\varphi_1(x_1,x_2,...x_n)+\\&+\lambda_2\varphi_2(x_1,x_2,...x_n)+...+\lambda_m\varphi_m(x_1,x_2,...x_n)=0,\end{aligned}\qquad(4.7.6)$$

по відповідних змінних $x_1,x_2,...x_n$. У виразі (4.7.6) $\lambda_1,\lambda_2,...\lambda_m$ називаються множниками Лагранжа. Прирівнюючи нулю виразу для відповідних приватних похідних, отримаємо

$$\begin{aligned}
&\frac{\partial f(x_1,x_2,...x_n)}{\partial x_1}+\lambda_1\frac{\partial \varphi_1(x_1,x_2,...x_n)}{\partial x_1}+\lambda_2\frac{\partial \varphi_2(x_1,x_2,...x_n)}{\partial x_1}+...+\\
&\qquad+\lambda_m\frac{\partial \varphi_m(x_1,x_2,...x_n)}{\partial x_1}=0,\\
&\frac{\partial f(x_1,x_2,...x_n)}{\partial x_2}+\lambda_1\frac{\partial \varphi_1(x_1,x_2,...x_n)}{\partial x_2}+\lambda_2\frac{\partial \varphi_2(x_1,x_2,...x_n)}{\partial x_2}+...+\\
&\qquad+\lambda_m\frac{\partial \varphi_m(x_1,x_2,...x_n)}{\partial x_2}=0,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\\
&\frac{\partial f(x_1,x_2,...x_n)}{\partial x_n}+\lambda_1\frac{\partial \varphi_1(x_1,x_2,...x_n)}{\partial x_n}+\lambda_2\frac{\partial \varphi_2(x_1,x_2,...x_n)}{\partial x_n}+...+\\
&\qquad+\lambda_m\frac{\partial \varphi_m(x_1,x_2,...x_n)}{\partial x_n}=0.
\end{aligned}\tag{4.7.7}$$

Вирішуючи спільно $m+n$ рівнянь (4.7.5), (4.7.7) визначаємо значення $x_{10},x_{20},...x_{n0}$ змінних, при яких функція $f(x_1,x_2,...x_n)$ досягає умовного екстремуму і значення множників Лагранжа $\lambda_1,\lambda_2,...\lambda_m$.

Для визначення достатніх умов існування умовного екстремуму поступають аналогічно, як і у разі незалежних змінних $x_1,x_2,...x_n$. Проте, функцію $f(x_1,x_2,...x_n)$ необхідно замінити функцією (4.7.6). Так для функції два змінних x_1,x_2 отримаємо наступні умови: функція $f(x_1,x_2)$ має відносний максимум в крапці $M_0(x_{10},x_{20})$ якщо

$$\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2>0,\quad \frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1^2}<0,$$

функція $f(x_1,x_2)$ має відносний мінімум, якщо

$$\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2>0,\quad \frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1^2}>0,$$

функція $f(x_1,x_2)$ не має ні максимуму, ні мінімуму, якщо

$$\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1^2}\cdot\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_2^2}-\left[\frac{\partial^2\Phi(x_{10},x_{20})}{\partial x_1\partial x_2}\right]^2<0,$$

функція $f(x_1,x_2)$ може мати екстремум в крапці $M_0(x_{10},x_{20})$а може і не мати, якщо

$$\frac{\partial^2 \Phi(x_{10}, x_{20})}{\partial x_1^2} \cdot \frac{\partial^2 \Phi(x_{10}, x_{20})}{\partial x_2^2} - \left[\frac{\partial^2 \Phi(x_{10}, x_{20})}{\partial x_1 \partial x_2}\right]^2 = 0.$$

У разі, коли змінні зв'язані додатковими умовами $\sum_{i=1}^{m} \varphi_i(x_1, x_2, \ldots x_n) = 0$ у вигляді лінійних функцій і нерівностей, що містять увігнуті функції, то можна використовувати умови Куна-Таккера для загального завдання оптимізації у разі нелінійного програмування з обмеженнями у вигляді рівності і нерівностей [38].

Розглянемо загальне завдання нелінійного програмування, коли необхідно мінімізувати функцію $f(x_1, x_2, \ldots x_n)$ при наступних обмеженнях

$$\sum_{i=1}^{m} \varphi_i(x_1, x_2, \ldots x_n) = 0 \quad \sum_{j=1}^{V} g_j(x_1, x_2, \ldots x_n) \geq 0$$

Достатність умов Куна-Таккера полягає в наступному. Якщо цільова функція $f(x_1, x_2, \ldots x_n)$ опукла, а всі обмеження у вигляді нерівностей $\sum_{j=1}^{V} g_j(x_1, x_2, \ldots x_n) \geq 0$ містять увігнуті функції $g_j(x_1, x_2, \ldots x_n)$ а обмеження у вигляді рівності містять лінійні функції $\varphi_i(x_1, x_2, \ldots x_n)$ то якщо існує рішення $(x_{10}, x_{20}, \ldots x_{n0}, u_1, u_2 \ldots u_V, \lambda_1, \lambda_2, \ldots \lambda_m)$ що задовольняє умовам

$$\frac{\partial f(x_1, x_2, \ldots x_n)}{\partial x_k} - \frac{\partial}{\partial x_k}\left[\sum_{k_1=1}^{V} u_{k_1} g_{k_1}(x_1, x_2, \ldots x_n)\right] - \frac{\partial}{\partial x_k}\left[\sum_{k_2=1}^{m} \lambda_{k_2} \varphi_{k_2}(x_1, x_2, \ldots x_n)\right] = 0,$$

$$k = 1,2,\ldots n, \quad k_1 = 1,2,\ldots V, \quad k_2 = 1,2,\ldots m,$$

$$\sum_{i=1}^{m} \varphi_m(x_1, x_2, \ldots x_n) = 0, \quad \sum_{j=1}^{V} g_j(x_1, x_2, \ldots x_n) \geq 0, \qquad (4.7.8)$$

$$u_{k_1} g_{k_1}(x_1, x_2, \ldots x_n) = 0, \quad u_{k_1} \geq 0,$$

то сукупність $x_{10}, x_{20}, \ldots x_{n0}$ є оптимальним рішенням задачі нелінійного програмування [38].

У разі, коли цільова функція $f(x_1, x_2, \ldots x_n)$ є лінійною функцією змінних $x_1, x_2, \ldots x_n$ а умови $\sum_{i=1}^{m} \varphi_i(x_1, x_2, \ldots x_n) = 0 \quad \sum_{j=1}^{V} g_j(x_1, x_2, \ldots x_n) \geq 0$

$f(x_1, x_2, \ldots x_n)$.

Класичними завданнями лінійного програмування є: транспортне завдання (необхідно так спланувати перевезення, щоб мінімізувати витрати, пов'язані з транспортуванням вантажів); завдання про використання сировини (при фіксованих запасах сировини необхідно так спланувати випуск декількох видів продукції, щоб прибуток підприємства був максимальним); завдання про використання устаткування (необхідно скласти план використання декількох видів виробничого устаткування з різною продуктивністю, щоб можна було виконати план виробництва по номенклатурі виробів); завдання проектування механізмів, пристроїв і систем механізмів (потрібно спроектувати механізм так, щоб його надійність була найбільшою, а маса, габарити і вартість менше деяких наперед заданих величин).

Суть завдань лінійного програмування полягає у відшуканні ненегативних значень змінних $x_{10}, x_{20}, \ldots x_{n0}$ які забезпечують досягнення найбільшого(найменшого) значення цільової функції $f(x_1, x_2, \ldots x_n)$. Завдання лінійного програмування можуть містити обмеження змінних $x_1, x_2, \ldots x_n$ у вигляді рівності. Тоді отримуємо основне завдання лінійного програмування

$$f(x_1, x_2, \ldots x_n) = b_{11}x_1 + b_{12}x_2 + \ldots + b_{1n}x_n + b_0, \qquad (4.7.9)$$

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = c_m, \\ m < n, \end{gathered} \qquad (4.7.10)$$

де $b_0, b_{11}, \ldots b_{1n}, a_{11}, \ldots a_{mn}, c_1, \ldots c_m$ - деякі постійні.

Будь-яким допустимим рішенням основної задачі лінійного програмування можна вважати сукупність змінних $x_1, x_2, \ldots x_n$ яка задовольняє наступним нерівностям

$$x_1 \geq 0, x_2 \geq 0, \ldots x_n \geq 0. \quad (4.7.11)$$

Крім того, необхідно, щоб вказана сукупність змінних $x_1, x_2, \ldots x_n$ задовольняла системі рівнянь (4.7.10). Оптимальним рішенням, зі всіх допустимих рішень, буде таке, при якому значення цільової функції (4.7.9) прийме мінімальне (максимальне) значення. У разі коли сукупність змінних $x_1, x_2, \ldots x_n$ задовольняє (4.7.10), але не задовольняє (4.7.11) або рівняння системи (4.7.10) суперечать один одному, то говорять, що завдання лінійного програмування не має допустимих рішень. Коли цільова функція $f(x_1, x_2, \ldots x_n)$ в області допустимих рішень не обмежена знизу, то серед допустимих рішень основної задачі лінійного програмування немає оптимального рішення.

У разі, коли обмеження, що визначають допустимі значення змінних $x_1, x_2, \ldots x_n$ задані у вигляді лінійних нерівностей, то говорять про завдання лінійного програмування з обмеженнями-нерівностями

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - c_1 \geq 0, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - c_2 \geq 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n - c_m \geq 0. \end{aligned} \quad (4.7.12)$$

Використовуючи (4.7.12) необхідно знайти таку сукупність ненегативних значень змінних $x_1, x_2, \ldots x_n$ щоб набути мінімального (максимальне) значення цільової функції (4.7.9). Дану задачу вирішують шляхом переходу до основного завдання лінійного програмування шляхом введення нових базисних (додаткових) змінних

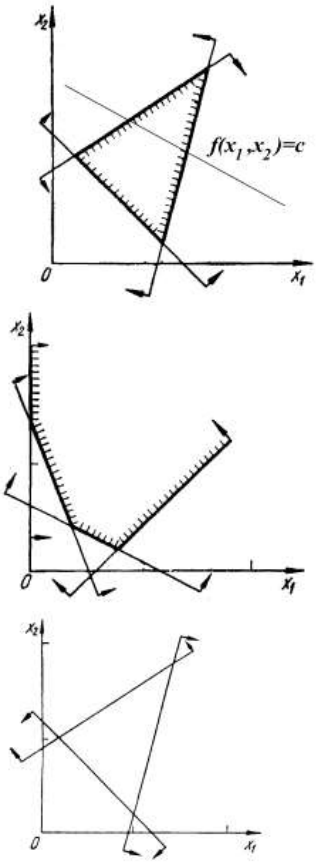
$$\begin{aligned} y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - c_1, \\ y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - c_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_m = a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n - c_m. \end{aligned} \quad (4.7.13)$$

Базисні змінні $y_1, y_2, \ldots y_m$ і початкові $x_1, x_2, \ldots x_n$ повинні задовольняти нерівностям

$$y_j \geq 0, x_k \geq 0, \quad j = 1, \ldots m, k = 1, \ldots n. \quad (4.7.14)$$

Отже, необхідно знайти не негативні значення $n+m$ змінних $x_1, x_2, \ldots x_n$ $y_1, y_2, \ldots y_m$

Геометрична інтерпретація системи лінійних нерівностей (4.7.12) (залежно від величини коефіцієнтів $a_{11}, \ldots a_{mn}, c_1, \ldots c_m$) може бути представлена трьома варіантами (для наочності обмежимося двома змінними x_1 і x_2), які представлені на рис.4.7.1 а-в.



а

б у

Рис.4.7.1. Області вирішень системи лінійних нерівностей

Рис.4.7.1 а показує, що областю вирішення системи нерівностей (4.7.12) є обмежене, опукле тіло (трикутник). Стрілками на малюнку вказані на пів площини, які є областями вирішень відповідних нерівностей системи (4.7.12), яка є визначеною. На рис.4.7.1 би показаний випадок, коли область вирішення системи нерівностей (4.7.12) є необмеженим, опуклим тілом. В даному випадку система нерівностей є невизначеною. На рис.4.7.1 в область вирішень системи нерівностей (4.7.12) порожня – не містить жодної крапки. Дана система нерівностей є несумісною.

Для пошуку оптимального рішення в завданні лінійного програмування розглянемо лінії рівня функції (4.7.9) на площині x_1, x_2 (см.рис.4.7.1а)

$$b_{11}x_1 + b_{12}x_2 = c. \qquad (4.7.15)$$

Рівняння (4.7.15) описує сімейство прямих, паралельних прямою

$$b_{11}x_1 + b_{12}x_2 = 0. \qquad (4.7.16)$$

Оптимальне значення (максимальне або мінімальне) цільової функції відповідатиме прямій сімейства (4.7.15), яка міститиме крайню крапку з області вирішення системи нерівностей (4.7.12). У разі пошуку максимуму пряма повинна бути найбільш видаленою від початку координат, а у разі пошуку мінімуму пряма повинна бути найменш видалена від початку координат.

Для вирішення завдань лінійного програмування з числом змінних $x_1, x_2, \ldots x_n$ більше двох використовується симплексний метод (метод послідовного поліпшення плану). Суть його полягає в тому, що знаючи деякий опорний план завдання, шляхом кінцевого числа кроків необхідно отримати оптимальний план виконуючи перехід від однієї крайньої точки опуклої множини до іншої з виконанням необхідної умови – зменшенням (збільшенням) значення цільової функції $f(x_1, x_2, \ldots x_n)$.

Одним з розділів математичного програмування є динамічне програмування – спеціальний обчислювальний метод оптимізації

вирішення завдань, в основі якого лежить принцип оптимальності що стверджує, що який би не був шлях досягнення деякого стану системи, подальші рішення повинні належати оптимальній програмі для частини шляху, що залишилася, починається з цього стану. Для застосування принципу оптимальності в конкретних завданнях користуються прийомом, який називають зануренням. Його суть полягає в тому, що замість рішення початкової задачі з даним початковим станом і даним числом кроків, вирішується ціле сімейство завдань з довільним початковим станом і з довільним числом кроків [14].

Припустимо, що деяка система S може змінювати свій стан. Дана система S буде керованою якщо ми зможемо впливати на стан даної системи переводячи її з одного стану в інше за допомогою деяких дій, які назвемо управлінням u. Критерій стану системи W є функцією управління

$$W = W(u). \tag{4.7.17}$$

Суть завдання оптимального управління полягає у відшуканні такого u^* яке дозволяє переводити систему S з початкового стану в кінцевий так, щоб критерій W приймав максимальне (мінімальне) значення

$$W^* = W(u^*) = \max_u [W(u)] \ . \tag{4.7.18}$$

Стан системи S характеризується певною кількістю параметрів $x_1, x_2, \ldots x_n$ (фазовими координатами системи). Процес переходу системи з початкового положення в кінцеве розіб'ємо на певне число кроків. Після цього, на кожному з кроків, починаючи з останнього, визначимо умовне оптимальне управління. Це робиться з урахуванням всіляких припущень про результати попереднього кроку. Дії продовжуються до тих пір, поки процес оптимізації буде доведений до початкового стану системи S. Потім знову проходиться вся послідовність кроків починаючи з першого. На кожному кроці з безлічі умовних оптимальних управлінь вибирається одне. Тут необхідно відзначити, що початковий стан системи S може бути

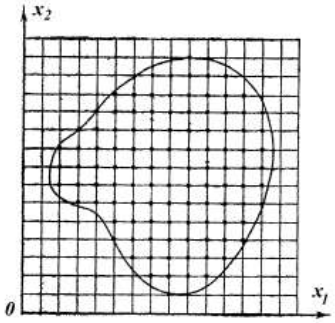
точно задано, або його необхідно вибрати з допустимої області з умовою досягнення максимуму (мінімуму) критерію W у початковому стані.

Рішення задачі оптимізації, з геометричної точки зору, полягає у відшуканні максимальної (мінімальною) точки поверхні відгуку цільової функції $f(x_1, x_2, ...x_n)$. У разі, коли вид цільової функції відомий екстремальні крапки визначаються по описаних вище методиках. У іншому випадку може бути, що аналітичної залежності цільової функції немає, а у дослідника є можливість визначити її значення в будь-якій точці даної області (за допомогою експериментальних досліджень, в результаті розрахунку). Очевидно, що це дозволяє визначити значення цільової функції лише в кінцевому числі крапок. Надалі, при описі методів пошуку екстремуму цільової функції $f(x_1, x_2, ...x_n)$ розглядатимемо функції двох параметрів $f(x_1, x_2)$.

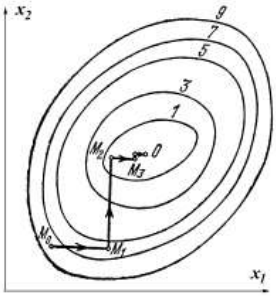
На рис.4.7.2 а показана реалізація методу, коли дана область зміни параметрів x_1, x_2 покривається сіткою з вертикальним і горизонтальним кроком h. Після цього, визначається значення цільової функції $f(x_1, x_2)$ у вузлах сітки і вибирається вузол, в якому маємо максимальне значення. Проте, даний метод є вельми наближеним і вимагає значних витрат машинного часу для розрахунків.

На рис.4.7.2 б показана реалізація методу по координатного спуску. Його суть полягає в наступному. На першому етапі в початковій точці M_0 фіксуємо змінну x_2 і починаємо змінювати x_1. Рухатимемося від початкової точки $x_1 = x_{10}$ у бік убування (зростання) функції, поки не дійдемо до її мінімуму $x_1 = x_{11}$ (максимуму)після якого вона починає зростати (убувати). Дану крапку позначимо M_1. Фіксуємо тепер змінну $x_1 = x_{11}$ і починаємо змінювати другу змінну $x_2 = x_{20}$ у бік убування (зростання) цільової функції. Знову дійдемо до мінімуму (максимуму) цільової функції. Позначимо цю крапку через M_2. Після цього знову

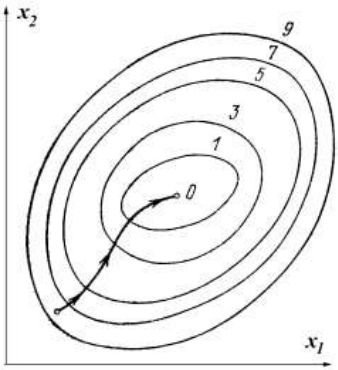
повертаємося до змінній x_1. Даний процес обриваємо на деякому кроці і приблизно приймаємо значення цільової функції за екстремальне.

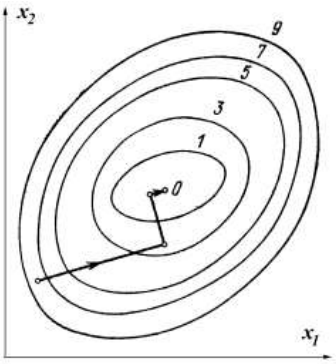


а

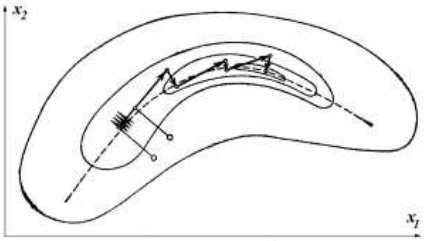


б

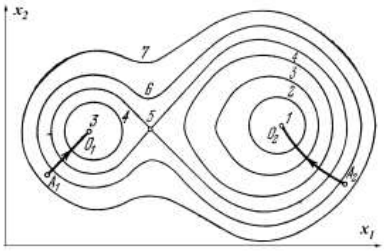




г



д



е

Рiс.4.7.2. Методи визначення
екстремуму цільової функції $f(x_1, x_2)$

На рис.4.7.2 в показана реалізація методу градієнтного спуску (підйому). Для цього методу необхідно знати вираз цільовій функції. Вектор градієнта показує напрям найбільш швидкого зростання функції. Його модуль рівний

$$\left| grad \quad f(x_1, x_2) \right| = \sqrt{\left[\frac{\partial f(x_1, x_2)}{\partial x_1}\right]^2 + \left[\frac{\partial f(x_1, x_2)}{\partial x_2}\right]^2}. \qquad (4.7.19)$$

Суть методу полягає в наступному. Вибираємо початкову точку і визначаємо в ній градієнт. Після цього робимо невеликий крок в цьому напрямі. У новій крапці повторюємо процедуру. Складність даного методу полягає у визначенні по (4.7.19) вирази для градієнта на кожному кроці.

Досконалішою є методика модифікованого градієнтного методу – метод найскорішого спуску. На рис.4.7.2 г показана його реалізація. Послідовність операцій буде наступна. У початковій точці визначають градієнт і роблять в його напрямі не маленький крок, а рухаються в його напрямі до тих пір, поки цільова функція зростає. У цій крапці знову обчислюють градієнт і рухаються в новому напрямі.

Вельми часто при пошуку екстремуму цільової функції дослідники стикаються з проблемою «ярів» (лінії рівня сильно витягнуті в одному напрямі і сплюснуті в іншому). На рис.4.7.2 д показані перетини поверхні відгуку з областю «яру». Для вирішення цієї проблеми з двох близьких крапок здійснюють градієнтний спуск на дно «яру». Потім сполучають знайдені точки прямої і роблять уздовж неї великий крок «яру». З отриманої крапки знову здійснюють градієнтний спуск на дно «яру» і роблять наступний крок «яру».

В деяких випадках цільові функції володіють властивістю багато екстремальності (см.рис.4.7.2 е). Для визначення абсолютного екстремуму необхідно проводити пошук з декількох різних крапок. Якщо набудемо різних значень цільової функції, то після їх порівняння вибираємо найбільше (найменше).

Приклад № 1. Завдання пошуку оптимального розміру циліндрової судини. Необхідно визначити радіус r підстави судини постійного об'єму V. Як цільові функції, на попередньому етапі, виберемо дві. Зажадаємо щоб площа поверхні циліндрової судини $S(r)$ була мінімальною (на його

виготовлення піде менше матеріалу) і зажадаємо щоб довжина зварних швів $l(r)$ була мінімальною (економиться час на виконання операції зварювання при виготовленні судини). Вирази для цільових функцій мають вигляд

$$\begin{aligned} S(r) &= 2\pi r^2 + 2\pi rh, \\ l(r) &= 4\pi r + h, \\ V &= \pi r^2 h, \end{aligned} \qquad (4.7.20)$$

де h - висота судини.

Перетворимо (4.7.20) до вигляду

$$\begin{aligned} S(r) &= 2\pi r^2 + \frac{2V}{r}, \quad 0 < r < \infty, \\ l(r) &= 4\pi r + \frac{V}{\pi r^2}, \quad 0 < r < \infty. \end{aligned} \qquad (4.7.21)$$

Набутих значень цільових функцій показую, що вони залежать від однієї змінної r. Скориставшись (4.7.2) визначимо їх похідні і прирівняємо нулю

$$\begin{aligned} \frac{\partial S(r)}{\partial r} &= \frac{2}{r^2}(2\pi r^3 - V) = 0, \\ \frac{\partial l(r)}{\partial r} &= \frac{2}{\pi r^3}(2\pi^2 r^3 - V) = 0. \end{aligned}$$

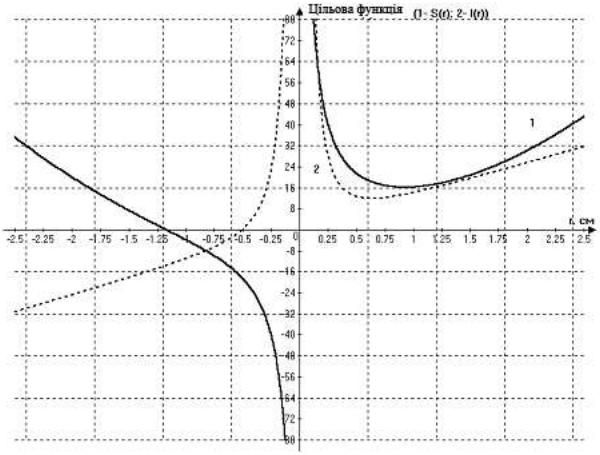
З отриманої системи рівнянь визначаємо значення радіусу циліндрової судини при якому отримаємо мінімальну площу поверхні $S(r_S)$ і мінімальну довжину швів $l(r_l)$

$$\begin{aligned} r_S &= \sqrt[3]{\frac{V}{2\pi}}, \quad h_S = 2r_S, \\ r_l &= \sqrt[3]{\frac{V}{2\pi^2}}, \quad h_l = 2\pi r_l. \end{aligned} \qquad (4.7.22)$$

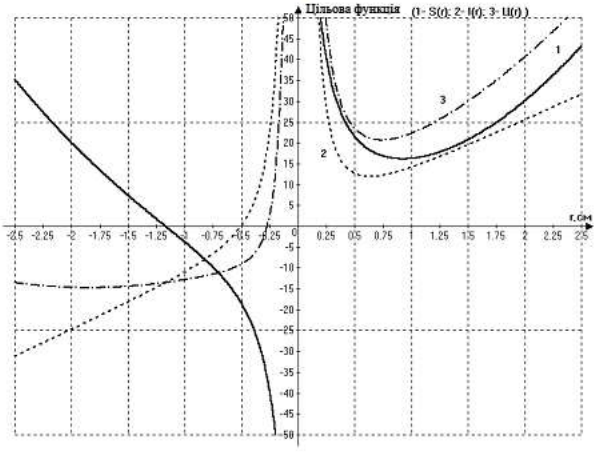
На рис.4.7.3 а показані графічні залежності зміни цільових функцій $S(r)$ і $l(r)$ (при розрахунках приймали V=5 см3).

Ускладнимо завдання. Хай нам необхідно визначити радіус r циліндрової судини постійного об'єму V. Нам вимагається щоб на виготовлення судини пішов якомога менше матеріалу, щоб довжина зварних швів була мінімальною. Крім того, відомо що ціна одиниці площі

матеріалу судини $Ц_1$ а ціна одиниці довжини зварного шва $Ц_2$. Тоді як цільова функція можна узяти вартість циліндрової судини $Ц$.



а



б

Рис.4.7.3. Графіки зміни цільових функцій $S(r)$, $l(r)$ и $Ц(r)$

З обліком (4.7.20), (4.7.21), отримаємо

$$Ц(r) = 2Ц_1 \pi r^2 + \frac{2Ц_1 V}{r} + 4Ц_2 \pi r + \frac{VЦ_2}{\pi r^2}, \quad 0 < r < \infty. \tag{4.7.23}$$

Визначимо першу похідну і прирівняємо її нулю

$$\frac{\partial Ц(r)}{\partial r} = 4Ц_1 \pi r - \frac{2Ц_1 V}{r^2} + 4Ц_2 \pi - \frac{2VЦ_2}{\pi r^3} = 0. \tag{4.7.24}$$

Після перетворень (4.7.24) прийме вигляд

$$4Ц_1 \pi^2 r^4 - 2Ц_1 V \pi r + 4Ц_2 \pi^2 r^3 - 2VЦ_2 = 0.$$

Вирішуємо отримане рівняння четвертого ступеня чисельним методом (см.4.7). Для V=5см^3, $Ц_1$=0,5 ед. ціни $Ц_2$=1 ед. ціни отримаємо чотири корені: r_1=0,7364 см r_2 = - 1,8472 см $r_{3,4}$ = - 0,4446 ±0,418j . Комплексне коріння відкидаємо і внаслідок того, що $0 < r < \infty$ отримуємо $r_Ц$ =0,7364 див.

На рис.4.7.3 би показані графічні залежності зміни цільових функцій $S(r)$ $l(r)$

$$r_l < r_Ц < r_S .$$

Приклад № 2. Визначення мінімальної відстані r_0 від носика спрямовувача нитки (з координатами $x = a, y = b, z = c$) до відбійної площини плосков'язальної машини, вираженої рівнянням

$$Ax + By + Cz + D = 0, \tag{4.7.25}$$

де A, B, C, D - деякі постійні.

Квадрат відстані від носика спрямовувача нитки до змінної крапки (x, y, z) виражається залежністю

$$r^2 = (x-a)^2 + (y-b)^2 + (z-c)^2 . \tag{4.7.26}$$

Вирішуємо дану задачу, з обмеженнями у вигляді рівності, з використанням множників Лагранжа. Необхідно мінімізувати вираз (4.7.26). Допоміжна функція має вигляд (з обліком (4.7.6))

$$\Phi = (x-a)^2 + (y-b)^2 + (z-c)^2 + \lambda_1 (Ax + By + Cz + D).$$

Визначаємо її приватні похідні і прирівнюємо їх нулю. Тоді отримаємо

$$x = a - \frac{1}{2}\lambda_1 A,$$
$$y = b - \frac{1}{2}\lambda_1 B, \qquad (4.7.27)$$
$$z = c - \frac{1}{2}\lambda_1 C.$$

Підставляючи значення x, y, z у (4.7.25) визначаємо значення множника Лагранжа

$$\lambda_1 = \frac{2(Aa + Bb + Cc + D)}{A^2 + B^2 + C^2}. \qquad (4.7.28)$$

Підставляючи (4.7.27) в (4.7.26), з обліком (4.7.28), отримаємо вираз для визначення мінімальної відстані від носика спрямовувача нитки до відбійної площини плосков'язальної машини

$$r_0^2 = \frac{1}{4}\lambda_1^2 (A^2 + B^2 + C^2).$$

Приклад № 3. Проектування коробки у формі паралелепіпеда з шматка матеріалу постійної площі $2a$ найбільшого об'єму V. Цільова функція матиме вигляд

$$V = xyz, \qquad (4.7.29)$$

де x, y, z - довжини сторін паралелепіпеда.

Рівняння, що зв'язує змінні x, y, z має вигляд

$$xy + xz + yz - a = 0,$$
$$0 < x < \infty, \quad 0 < y < \infty, \quad 0 < z < \infty. \qquad (4.7.30)$$

Складаємо вираз для допоміжної функції з використанням множника Лагранжа

$$\Phi = xyz + \lambda(xy + xz + yz - a).$$

Визначаємо приватні похідні від останнього виразу і прирівнюємо їх нулю

$$yz + \lambda(y + z) = 0,$$
$$xz + \lambda(x + z) = 0, \qquad (4.7.31)$$
$$xy + \lambda(x + y) = 0.$$

Вирішуючи спільно (4.7.30) і (4.7.31), отримаємо

$$\lambda = -\frac{3xyz}{2a}.$$

Підставляючи λ у (4.7.31), отримаємо

$$\frac{3x}{2a}(y+z)=1,\quad \frac{3y}{2a}(x+z)=1,\quad \frac{3z}{2a}(y+x)=1.$$

Вирішуючи останню систему рівнянь визначимо, що $x=y=z=\sqrt{\frac{a}{3}}$. Таким чином, коробка матиме максимальний об'єм при постійній площі поверхні, коли вона формою представлятиме куб.

Література по розділу 1

1. Щербань В.Ю., Колиско О.З., Колиско М.І., Кириченко А.М., Щербань Ю.Ю. Комп'ютерні процедури програмного комплексу для визначення напруженості процесу подачі нитки на круглов'язальних машинах / В.Ю. Щербань, О.З. Колиско, М.І. Колиско, А.М. Кириченко, Ю.Ю. Щербань // Вісник Хмельницького національного університету. – 2022, №1 (305). – С. 256-259.

2. Щербань В.Ю., Колиско О.З., Колиско М.І., Кириченко А.М., Щербань Ю.Ю. Комп'ютерна реалізація алгоритму рекурсії для гребінчатого пристрою натягу панчішних автоматів / В.Ю. Щербань, О.З. Колиско, М.І. Колиско, А.М. Кириченко, Ю.Ю. Щербань // Вісник Хмельницького національного університету. – 2022, №2 (307). – С. 194-197.

3. Щербань В.Ю., Іщенко В. Д., Колиско О.З., Гольдберг М.І., Щербань Ю.Ю. Комп'ютерна реалізація алгоритму Дейкстри для визначення форми заправки нитки на основі пошуку оптимального шляху графа / В.Ю. Щербань, В. Д. Іщенко, О.З. Колиско, М.І. Гольдберг, Ю.Ю. Щербань // Вісник Хмельницького національного університету. – 2022, №3 (309). – С. 217-220.

4. Щербань В.Ю., Іщенко В. Д., Колиско О.З., Гольдберг М.І., Щербань Ю.Ю. Визначення вагових функцій ребер неорієнтованого графа при комп'ютерному пошуку оптимального шляху з використанням алгоритму Дейкстри / В.Ю. Щербань, В. Д. Іщенко, О.З. Колиско, М.І. Гольдберг, Ю.Ю. Щербань // Вісник Хмельницького національного університету. – 2022, №4 (311). – С. 270-273.

5. Vasilchenko V.N. Steady motion of a textile yarn with two anchoring points over a rough surface / V.N. Vasilchenko, V.Yu. Shcherban, Ts.V. Apokin // Technology of the textile industry. - 1985. - № 4. - P.54-56.

6. Vasilchenko V.N. Equilibrium of a filament of a root base in the zone of formation of a multilayer technical fabric / V.N. Vasilchenko, V.Yu. Shcherban // Technology of the textile industry. - 1986. - № 5. - P.44-47.

7. Vasilchenko V.N. Influence of the twist of a capron complex filament on the value of its flexural rigidity / V.N. Vasilchenko, V.Yu. Shcherban // Technology of the textile industry. - 1986. - №4. - P.8-9.

8. Scherban V.Yu. Determination of the geometric characteristics of the shape of the filament axis moving along the deformable guide surface / V.Yu. Shcherban // Technology of the textile industry. - 1990. - №6. - P.52-55.

9. Yakubitskaya I.A. Dynamic analysis of layout conditions on the end sections of the groove of the winding drum / I.A. Yakubitskaya, V.V. Chugin, V.Yu. Shcherban // Technology of the textile industry. - 1997. - №5. - P.33-37.

10. Ресурсоощадні технології виробництва текстилю, одягу та взуття: монографія: в 2 т. Т.1/Теоретичні основи та методи розроблення ресурсоощадних технологій та обладнання для виробництва текстилю, одягу та взуття/ В.Ю.Щербань, Б.Ф.Піпа, В.В.Чабан та ін. – К.:КНУТД, 2016. – 373 с.

11. Vasil'chenko V.N., Shcherban' V.Yu., Apokin Ts.V. Attachment for holding multilayer fabrics in the clamps of a universal tensile tester/ V.N.Vasil'chenko , V.Yu.Shcherban' , Ts.V.Apokin // Textile industry. – 1987. - №8. - pp.62.

12. Shcherban’ V., Makarenko J., Melnyk G., Shcherban’ Y., Petko A., Kirichenko A. Effect of the yarn structure on the tension degree when interacting with high-curved guides/ V. Shcherban’, J. Makarenko, G. Melnyk, Y. Shcherban’, A. Petko, A. Kirichenko // Fibres and Textiles. – 2019. - volume 26 - № 4 - pp. 59-68.

13. Shcherban’ V., Makarenko J., Petko A., Melnyk G., Shcherban’ Yu., Shchutska G. Computer implementation of a recursion algorithm for determining the tension of a thread on technological equipment based on the

derived mathematical dependences / V.Shcherban', J.Makarenko, A.Petko, G.Melnyk, Yu.Shcherban', G.Shchutska // Eastern-European Journal of Enterprise Technologies. - 2020. - volume 104. -№2/1. – pp.41-50.

14. Shcherban' V., Kolysko O., Melnyk G., Sholudko M., Shcherban' Y. and Shchutska G. Determining tension of yarns when interacting with guides and operative parts of textile machinery having the torus form / V. Shcherban', O. Kolysko, G. Melnyk, M. Sholudko, Y. Shcherban' and G. Shchutska // Fibres and Textiles. – 2020. - volume 27 - № 4 - pp. 87-95.

15. Shcherban' V., Korogod G., Kolysko O., Kolysko M., Shcherban' Yu., Shchutska G. Computer simulation of multiple measurements of logarithmic transformation function by two approaches / V. Shcherban', G. Korogod, O. Kolysko, M. Kolysko, Yu. Shcherban', G. Shchutska // Eastern-European Journal of Enterprise Technologies. - 2020. - volume 6. -№4 (108). – pp. 6-13.

16. Shcherban' V., Korogod G., Kolysko O., Kolysko M., Shcherban' Yu., Shchutska G. Computer simulation of logarithmic transformation function to expand the range of high-precision measurements / V. Shcherban', G. Korogod, O. Kolysko, M. Kolysko, Yu. Shcherban', G. Shchutska // Eastern-European Journal of Enterprise Technologies. - 2021. - volume 2. -№9 (110). – pp. 27-36.

17. Ресурсоощадні технології виробництва текстилю, одягу та взуття: монографія: в 2 т. Т.2/Підвищення надійності ресурсоощадних виробництв текстилю, одягу і взуття на основі новітніх технологій та системного управління/ В.Ю.Щербань, Б.Ф.Піпа, В.В.Чабан та ін. – К.:КНУТД, 2016. – 214 с.

18. Ресурсоощадні технології та обладнання швейної та текстильної промисловості: монографія: в 2 ч. Ч.1/Наукові основи та інженерні методи проектування ресурсоощадних технологій і обладнання швейної та текстильної промисловості/ В.Ю.Щербань, Г.Б.Параска, Б.В.Орловський та ін. – К.:КНУТД, 2015. – 339 с.

19. Ресурсоощадні технології та обладнання швейної та текстильної промисловості: монографія: в 2 ч. Ч.2/Шляхи підвищення ефективності швейної та текстильної галузей України на базі новітних технологій та управління/ В.Ю.Щербань, Г.Б.Параска, Б.В.Орловський та ін. – К.:КНУТД, 2015. – 270 с.

20. Прогнозування фізико-механічних властивостей текстильних матеріалів побутового призначення/А.М. Слізков , В.Ю. Щербань ,С.М. Краснитський , О.Б. Демківський. –К.:КНУТД, 2013. – 223 с.

21. Щербань В. Ю.Інформаційні технології в науці, виробництві та підприємництві/В.Ю.Щербань.-К.:КНУТД, 2016. – 184 с.

22. Щербань В. Ю.Інформаційні технології в науці, виробництві та підприємництві/В.Ю.Щербань.-К.:Освіта України, 2017. – 238 с.

23. Щербань В.Ю. Алгоритмічні, програмні та математичні компоненти САПР в індустрії моди/ В.Ю.Щербань, О.З.Колиско, М.І.Шолудько, В.Ю.Калашник. – К.:Освіта України, 2017. – 745 с.

24. Прогнозування процесів на основі моделювання часових рядів: навч. Посіб./П.І.Бідюк, В.Ю.Щербань, Є.О.Демківський, Т.І.Демківська.-К.:КНУТД, 2017.-324 с.

25. Щербань В.Ю. Математичні та програмні компоненти САПР технологічних процесів та обладнання текстильної та взуттєвої галузі/ В.Ю.Щербань, О.З.Колиско, М.І.Шолудько, В.Ю.Калашник. – К.:Бумсервіс, 2016. – 588 с.

26. Слізков А.М., Щербань В.Ю., Кизимчук О.П. Механічна технологія текстильних матеріалів. Частина ІІ. (Ткацьке, трикотажне та неткане виробництво): підручник / А.М.Слізков, В.Ю.Щербань, О.П.Кизимчук. – К.:КНУТД, 2018. – 276 с.

27. Щербань В.Ю. Механіка нитки/В.Ю.Щербань. – К.:Видавництво «Укрбланковидав». – 2018. – 533 с. https://er.knutd.edu.ua/handle/123456789/9517

28. Щербань В. Ю.Інформаційні технології в науці, виробництві та підприємництві/В.Ю.Щербань.-К.:Освіта України, 2018. – 257 с.

29. Щербань В.Ю. Базове проектуюче забезпечення САПР в індустрії моди/ В.Ю.Щербань, Ю.Ю.Щербань, О.З.Колиско, Г.В.Мельник, М.І.Шолудько, В.Ю.Калашник. – К.:Освіта України, 2018. – 902 с.

30. Щербань В. Ю. Інформаційні технології в науці, виробництві та підприємництві/В.Ю.Щербань.-К.:Освіта України, 2019. – 252 с.

31. Щербань В.Ю. Комп'ютерне проектування систем: програмні та алгоритмічні компоненти / В.Ю.Щербань, О.З.Колиско, Г.В.Мельник, М.І.Шолудько, В.Ю.Калашник. – К.:Освіта України, 2019. – 902 с.

32. Щербань В. Ю. Інформаційні технології в науці, виробництві та підприємництві/В.Ю.Щербань – К.:Освіта України: ФОП Маслаков, 2020. – 236 с.

33. Щербань В. Ю. Інформаційні технології в науці, виробництві та підприємництві / В.Ю.Щербань – К.:Освіта України, 2021. – 248 с.

34. Щербань В.Ю. Алгоритмічне та математичне забезпечення при комп'ютерному проектуванні складних систем / В.Ю.Щербань, О.З.Колиско, Ю.Ю.Щербань, Г.В.Мельник, М.І.Колиско, В.Ю.Калашник. – К.: Освіта України, 2021. – 930 с.

35. Scherban V.Yu. Determination of technological efforts in the process of surf during the formation of multilayer technical fabric / V.Yu. Shcherban // Technology of the textile industry. - 1990. - №3. - P.44-47.

36. Scherban V.Yu. Investigation of the process of duck surf during the formation of multilayer technical fabric / V.Yu. Shcherban // Technology of the textile industry. - 1990. - №4. - P.41-44.

37. Yakubitskaya I.A. Differential equations of the relative motion of the filament element on the end sections of the coil of the winding drum / I.A. Yakubitskaya, V.V. Chugin, V.Yu. Shcherban // Technology of the textile industry. - 1997. - №6. - P.50-54.

38. Shcherban' V.Yu. Interaction of stiff yarns with the working parts of knitting and sewing machines/V.Yu.Shcherban' // Textile industry. -1988. - № 10. - pp.53.

39. Shcherban' V., Melnyk G. , Sholudko M. and Kalashnyk V. Warp yarn tension during fabric formation/V.Shcherban' , G.Melnyk , M.Sholudko, V.Kalashnyk // Fibres and Textiles. – 2018. – volume 25. - №2. – pp.97-104.

40. Shcherban' V., Melnyk G. , Sholudko M., Kolysko O. and Kalashnyk V. Yarn tension while knitting textile fabric/V.Shcherban' , G. Melnyk , M.Sholudko , O.Kolysko, V.Kalashnyk// Fibres and Textiles. – 2018. - volume 25. - №3. - pp. 74-83.

41. Shcherban' V., Melnyk G. , Sholudko M., Kolysko O. and Kalashnyk V. Improvement of structure and technology of manufacture of multilayer technical fabric/V.Shcherban' , G. Melnyk , M.Sholudko , O.Kolysko, V.Kalashnyk// Fibres and Textiles. – 2019. - volume 26 - № 2 - pp. 54-63.

42. Shcherban' V., Korogod G., Chaban V., Kolysko O., Shcherban' Yu., Shchutska G. Computer simulation methods of redundant measurements with the nonlinear transformation function / V. Shcherban', G. Korogod, V. Chaban, O. Kolysko, Yu. Shcherban', G. Shchutska // Eastern-European Journal of Enterprise Technologies. - 2019. - volume 98. -№2/5. – pp.16-22.

43. Shcherban' V., Kolysko O., Melnyk G., Sholudko M., Shcherban' Yu., Shchutska G. and Kolva N. Determination of tension for polyamide and basalt multifilament yarns while weaving industrial fabrics / V. Shcherban', O. Kolysko, G. Melnyk, M. Sholudko, Yu. Shcherban', G. Shchutska, N. Kolva // Fibres and Textiles. – 2021. - volume 28 - № 1 - pp. 75-85.

44. Щербань В.Ю. Використання рекурсивного підходу для визначення натягу ниток в робочій зоні технологічного обладнання/В.Ю.Щербань, Н.І.Мурза, А.М. Кириченко, Г.В. Мельник, М.І.Шолудько//Вісник ХНУ.-2018.-№ 3(261). - С.7-11.

45. Щербань В.Ю. Взаємодія текстильних ниток з напрямними великої кривини у випадку наявності радіального охоплення/В.Ю.Щербань, Н.І.Мурза, А.М. Кириченко, Г.В. Мельник, М.І.Шолудько// Вісник Хмельницького національного університету.- 2018.-№ 2 (259). - С.12-16.

46. Щербань В.Ю. Удосконалення системи подачі ниток на основі оптимізації пружної системи заправки круглов'язальних машин/В.Ю.Щербань, Г.В. Мельник, Н.І.Мурза, А.М. Кириченко, М.І.Шолудько // Вісник Хмельницького національного університету.- 2018.-№ 4 (263). - С.11-16.

47. Щербань В.Ю. Структура комп'ютерної програми реалізації алгоритму

48. рекурсії для визначення технологічних зусиль/В.Ю. Щербань, А.К. Петко, О.З. Колиско, Ю.Ю. Щербань, М.І. Шолудько// Вісник Хмельницького національного університету.- 2020.-№ 1 (281). - С.249-253.

49. Первая Н.В., Андреєва О.А., Щербань В.Ю. Дослідження технологічних параметрів процесу формування верху взуття / Н.В. Первая, О.А. Андреєва, В.Ю. Щербань// Вісник Хмельницького національного університету.- 2020.-№ 1 (281). - С.175-181.

50. Щербань В.Ю. Програмні модулі комп'ютерної програми реалізації алгоритму рекурсії для випадку змінного вхідного натягу/ В.Ю.Щербань, А.К.Петко, О.З.Колиско, Ю.Ю.Щербань, М.І.Шолудько// Вісник Хмельницького національного університету.- 2020.-№ 2 (283). - С.213-218.

51. Щербань В.Ю., Петко А.К., Колиско О.З., Щербань Ю.Ю., Шолудько М.І. Комп'ютерна реалізація алгоритму рекурсії для випадку змінного діаметру сировини / В.Ю.Щербань, А.К.Петко, О.З.Колиско, Ю.Ю.Щербань, М.І.Шолудько // Вісник Хмельницького національного університету.- 2020, № 3(285). – С.263-267.

52. Щербань В.Ю., Колиско О.З., Щербань Ю.Ю., Шолудько М.І., Мельник Г.В. Алгоритмічні та програмні компоненти при комп'ютерному

визначенні натягу для шайбового натягувача з використанням рекурсії / В.Ю.Щербань, О.З.Колиско, Ю.Ю.Щербань, М.І.Шолудько, Г.В.Мельник // Вісник Хмельницького національного університету. Том 1.- 2020, № 4(287).– С.252-256.

53. Щербань В.Ю., Колиско О.З., Щербань Ю.Ю., Шолудько М.І., Мельник Г.В. Структура програмних модулів та процедур комп'ютерної програми для основних елементів системи при реалізації алгоритму рекурсії / В.Ю.Щербань, О.З.Колиско, Ю.Ю.Щербань, М.І.Шолудько, Г.В.Мельник// Вісник Хмельницького національного університету. - 2020, № 5(289).– С.302-306.

54. Щербань В.Ю., Петко А.К., Колиско О.З., Щербань Ю.Ю., Колиско М.І. База фрикційних властивостей комп'ютерної програми для визначення натягу нитки при реалізації алгоритму рекурсії / В.Ю.Щербань, А.К.Петко, О.З.Колиско, Ю.Ю.Щербань, М.І.Колиско // Вісник Хмельницького національного університету. - 2021, № 1(293).– С.234-237.

55. Щербань В.Ю., Петко А.К., Колиско О.З., Щербань Ю.Ю., Галавська Л.Є. Програмні модулі та процедури комп'ютерної програми для визначення натягу кевларової нитки при в'язанні з використанням алгоритму рекурсії / В.Ю.Щербань, А.К.Петко, О.З.Колиско, Ю.Ю.Щербань, Л.Є. Галавська // Вісник Хмельницького національного університету. - 2021, № 2(295).– С.271-274.

56. Щербань В.Ю., Макаренко Ю.В., Колиско О.З., Щербань Ю.Ю., Галавська Л.Є. Комп'ютерна реалізація алгоритму рекурсії при визначенні натягу ниток при формуванні багатошарових тканин з поліетиленових ниток / В.Ю. Щербань, Ю.В. Макаренко, О.З. Колиско, Ю.Ю. Щербань, Л.Є.Галавська // Вісник Хмельницького національного університету. – 2021, №3 (297). – С. 204-207.

57. Щербань В.Ю., Макаренко Ю.В., Колиско О.З., Щербань Ю.Ю., Колиско М.І. Реалізація програмних модулів процедури рекурсії при

комп'ютерному визначенні натягу основних ниток багатошарової тканини для військового спорядження / В.Ю. Щербань, Ю.В. Макаренко, О.З. Колиско, Ю.Ю. Щербань, М.І. Колиско // Вісник Хмельницького національного університету. – 2021, №4 (299). – С. 155-159.

2. ПРЕДСТАВЛЕННЯ ДАНИХ МОДЕЛЯМИ РЕГРЕСІЙНОГО ТА ДИСПЕРСІЙНОГО АНАЛІЗІВ

2.1. Про постановку задач регресійного аналізу

2.1.1 Попередні відомості

В багатьох експериментальних дослідженнях ми хочемо дізнатися, як зміни однієї величини впливають на іншу величину. Інколи дві змінні пов'язані точним функціональним співвідношенням. Наприклад, якщо опір R простого електричного ланцюга підтримується сталим, то струм I змінюється лінійно при лінійній зміні напруги V у відповідності із законом Ома $I = V/R$. Якби ми не знали закону Ома, то могли б знайти залежність емпірично, змінюючи V і вимірюючи R. Тоді ми побачили б, що графік залежності I від V наближено задається прямою лінією, що проходить через початок координат. Зазначена наближеність має місце з тієї причини, що наші виміри можуть містити певні похибки, і тому точки на графіку, скоріш за все, не розмістяться точно на одній прямій, хоча залежність фактично є точною. Іноді функціональна залежність принципово не є точною, навіть якщо не враховувати помилки. Наприклад, нехай розглядається зріст і вага дорослих чоловіків, обраних випадковим чином у деякій місцевості. Якщо ми нанесемо на графік пари чисел (Y_1,Y_2) = (зріст, вага), то результат у якісному відношенні буде відповідати наведеному нижче рис. 1.1. Відзначимо, що тут немає ніякого сенсу займатися підбором точної функціональної залежності між Y_1 та Y_2 (так щоб відповідний графік пройшов через всі експериментальні точки). І справа навіть не в тому, що така залежність була б надто складною. Більш істотне значення має та обставина, що при повторенні експерименту заново отримані експериментальні значення напевне не розташуються в точності на зазначеному графіку. Тож у даній ситуації слід підбирати

математичну модель зв'язку між змінними не в формі точної залежності, а в формі залежності наближеної, яка, не маючи надто складного вигляду, належним чином відтворила би найбільш суттєві тенденції зв'язку між досліджуваними змінними. Крім того, бажано було б одержати ще ймовірнісну оцінку похибки підібраної залежності. Математичні моделі зв'язків між змінними зазначеного типу часто називають *регресійними моделями* або *моделями регресії* (нижче будуть наведені уточнення поняття регресійної моделі, а також роз'яснення терміну "регресія" у даному контексті). Зауважимо, що у даному разі в якості такої моделі доцільно обрати залежність першого порядку (пряма лінія на рис. 1.1)

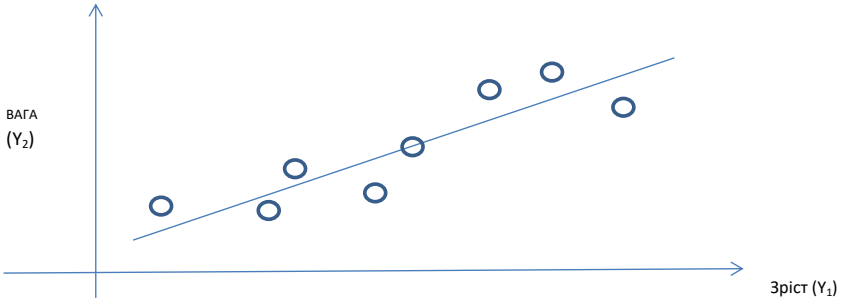


Рис. 1.1 Типова залежність між зростом і вагою дорослих чоловіків

Крім того, бажано було б одержати ще ймовірнісну оцінку похибки підібраної залежності. Математичні моделі зв'язків між змінними зазначеного типу часто називають *регресійними моделями* або *моделями регресії* (нижче будуть наведені уточнення поняття регресійної моделі, а також роз'яснення терміну "регресія" у даному контексті).

2.1.2 Одна загальна схема регресійного аналізу. Метод найменших квадратів

Нехай досліджується зв'язок між групою змінних $x_1,\ldots x_p$ та змінною y. При цьому вважається, що змінним $x_1,\ldots x_p$ можна надавати довільних значень (принаймні в певному діапазоні), а значення змінної y залежать від значень $x_1,\ldots x_p$ і одержуються експериментальним шляхом. Одна з найбільш загальних і важливих задач математичної статистики полягає у створенні математичної моделі, що описує (пояснює) вищезазначений зв'язок. Самі математичні моделі можуть бути різними. Наприклад, це можуть бути явні залежності, диференційні або інтегральні рівняння тощо. В регресійному аналізі застосовуються так звані *регресійні моделі* (*моделі регресії*). Однією з найпоширеніших регресійних моделей є модель у вигляді залежності

$$y = g(x_1,\ldots x_p;\, \beta_1,\ldots,\beta_m) + \varepsilon\,, \qquad (1.1)$$

в якій g – деяка функція, $\beta_1,\ldots,\beta_m$ – параметри залежності, ε – випадкова величина з нульовим математичним сподіванням. У відповідності із поширеною термінологією змінні $x_1,\ldots x_p$ звуться *незалежними змінними* або *регресорами*, змінна y – *залежною змінною або предиктором*, функція g – *функцією регресії*.

Основні задачі, що виникають при підборі (підгонці) регресійної залежності до експериментальних даних за допомогою моделі (1.1), формулюються наступним чином.

1. Обрати конкретну функцію g.
2. Оцінити (наближено визначити) параметри $\beta_1,\ldots,\beta_m$.
3. Перевірити адекватність (більш широко – можливість або доцільність застосування) обраної (побудованої) моделі.

4. Визначити ймовірнісні властивості випадкової складової ε, зокрема, оцінити її дисперсію (останнє – з метою визначення можливих відхилень залежної змінної $y = y(x_1,\ldots x_p)$ від значень функції регресії $g(x_1,\ldots x_p; \beta_1,\ldots,\beta_m)$).
5. Застосувати обрану модель для конкретних розрахунків і оцінити можливі похибки, що можуть виникнути при цьому.

   Перейдемо до обговорення можливостей розв'язку сформульованих задач.

1. Суто математичні методи відіграють лише допоміжну (хоча і важливу) роль в задачі вибору функції g. З повною надійністю такий вибір може бути зроблено лише за допомогою змістовних міркувань, що спираються на суть розглядуваного явища, його фізичний зміст. На жаль, досить часто зазначені міркування буває дуже важко навести, і тоді у першому наближенні задовольняються таким вибором функції g, який підказано графічним зображенням даних або діапазоном зміни незалежних змінних тощо. Якщо припустити, що вибір фунції g вже виконано, то можна перейти до другої з вищесформульованих задач.
2. Найчастіше оцінка невідомих параметрів у регресійному аналізі виконується за *методом найменших квадратів* (скорочено: МНК). Суть методу полягає в наступному. Нехай виконується n експериментів, в кожному з яких вектору незалежних змінних $x = (x_1,\ldots x_p)$ надається певних значень, і при цьому одержуються деякі значення залежної числової змінної y. Позначимо $x^i = (x^i_1,\ldots, x^i_p)$ набір значень назалежних змінних, що було надано їм в i-му експерименті, y_i – відповідні значення залежної змінної ($i = 1,2,\ldots, n$). Згідно з МНК в якості оцінки вектора параметрів $\beta = (\beta_1,\ldots,\beta_m)$ береться такий вектор $b = (b_1,\ldots, b_m)$ (інше позначення – $\hat{\beta} = (\hat{\beta}_1,\ldots,\hat{\beta}_m)$) при якому сума

$$S(\beta) = \sum_{i=1}^{n} \left[y_i - g(x^i, \beta) \right]^2 \qquad (1.2)$$

приймає мінімальне значення по $\beta \in R^m$, де R^m – m-вимірний евклідів простір.

Зауважимо, що сам принцип найменших квадратів в явному вигляді був сформульований ще у 18-му сторіччі видатними математиками Гауссом та Лежандром. Термін "регресійний аналіз" з'явився значно пізніше, вже у другій половині XIX сторіччя. Його пов'язують з роботами видатного англійського вченого Ф.Гальтона, який, зокрема, застосовував МНК в своїх антропологічних дослідженнях. При цьому Ф. Гальтон спостерігав деякі ознаки виродження, тобто *регресу*, певних розмірних людських ознак, так що в даних дослідженнях термін "регресія" був цілком виправданим. Роботи, про які йде мова, стали відомими, методика роботи автора з експериментальними даними була використана іншими дослідниками, і при цьому термін "регресія" автоматично зберігся і став вживатися без будь-яких обмежень на область досліджень та на характер закономірностей, що в цих дослідженнях виявляються.

На даний час розроблено багато ефективних обчислювальних алгоритмів для знаходження розв'язку задачі мінімізації виразу (1.2). На основі таких алгоритмів працюють відповідні комп'ютерні програми сучасного математичного забезпечення. Зауважимо, що у багатьох важливих випадках розв'язок зазначеної задачі можна одержати у явному вигляді.

Розв'язок задач 3 — 5 одержується на основі припущень щодо ймовірнісної поведінки величини ε – випадкової складової регресійної моделі. Найбільш фундаментальні результати у цьому напрямку на даний час одержано для

випадку, коли зазначена випадкова складова має нормальний розподіл ймовірностей. Деякі з цих результатів будуть наведені в подальшому.

2.2 Проста лінійна регресія — рівняння і оцінка параметрів

ВСТУП ДО ТЕМИ

В більшості експериментальних досліджень потрібно встановити, як перебіг значень одних змінних впливає на інші змінні. Іноді дві змінні зв'язані між собою лінійно. Деякі приклади такого зв'язку наводилися у попередньому підрозділі. Незалежно від того, має місце така залежність безпосередньо між змінними або лише для їх середніх значень (така ситуація зустрічається на практиці), знання її є вельми корисною. Можливо, така залежність в принципі не може бути точною, але доречні апроксимації її, можливо, на окремих ділянках однієї із змінних, також мають свою користь.

Зауважимо, що надалі, розглядаючи конкретні залежності «предикторної» або залежної змінної y від «регресору» або незалежної, інакше, пояснюючою змінною x, ми будемо вважати, що пояснююча змінна не є випадковою величиною. При цьому залежна змінна майже завжди вважається випадковою. З практичної точки зору пояснюючі змінні також можна було б вважати випадковими, але при цьому процедури підбору рівнянь значно ускладнюються. З приводу врахування можливої випадковості пояснюючих змінних автори рекомендують монографію [15].

2.2.1 Означення простої лінійної регресії і загальні зауваження

В попередньому розділі було відзначено, що, згідно з методом найменших квадратів (МНК), для оцінювання вектора параметрів $\beta = (\beta_1,\ldots,\beta_m)$ в регресійній моделі (5.1) треба розв'язати задачу мінімізації

$$S(\beta)=\sum_{i=1}^{n}\left[y_i-g(x^i,\beta)\right]^2\to min,\ \beta\in R^m, \tag{2.1}$$

тобто знайти таке $b\in R^m$, для якого значення функції $S(\beta)$ при $\beta = b$ є мінімальним (нагадаємо, що величини y_i та x^i при розв'язку задачі (2.1) слід вважати фіксованими).

Розв'язок задачі мінімізації (2.1) у даному підрозділі буде наведено для вельми часткового, але дуже важливого частинного випадку моделі (5.1). А саме, ми припустимо, що $p = 1$, отже вектор незалежних змінних x буде скалярною змінною. Далі вважатиметься, що $m = 2$. При цьому замість позначень β_1, β_2 для параметрів залежності буде використовуватися більш розповсюджене позначення β_0, β_1. Буде зроблене також дуже важливе припущення, що функція g є лінійною за параметрами β_0, β_1. Ми будемо ще вважати, що у виразі функції g змінна x присутня лише в степені 1. Отже, увага буде зосереджена на функції регресії g вигляду

$$g(x) = \beta_0 + \beta_1 x, \tag{2.2}$$

і, таким чином, досліджуватиметься наступний частковий випадок моделі (5.1):

$$y = \beta_0 + \beta_1 x + \varepsilon, \tag{2.3}$$

де, згідно з вищесказаним, x та y – відповідно, незалежна та залежна змінні, β_0, β_1 – параметри моделі, ε – випадкова складова моделі.

Залежність (2.3) має назву *простої лінійної регресії*.

2.2.1.1 Зауваження. Модель (2.3) може бути легко пристосованою до формально більш загальної ситуації, коли функція регресії має вигляд $g(x) = \beta_0 + \beta_1 \cdot u(x)$, де $u(x)$ – довільна (відома) функція від x. Дійсно, для зведення

такої моделі до (2.3) треба просто зробити заміну $u(x) = z$ і вважати z новою незалежною змінною.

2.2.2. Оцінювання параметрів моделі (2.3). Геометричний зміст МНК

Перейдемо безпосередньо до задачі оцінювання параметрів β_0, β_1 за експериментальними даними. Нехай незалежна змінна x в експериментах приймала значення $x_1,\ldots,x_n$ (останні позначення дещо відрізняються від застосованих вище – у розділі 1 $x_1,\ldots,x_n$ позначали різні незалежні змінні; тепер незалежна змінна тільки одна, при цьому $x_1,\ldots,x_n$ є її значеннями; в попередніх позначеннях довелося б писати $x_1^1,\ldots,x_n^1$), а залежна змінна y – відповідно, $y_1,\ldots,y_n$. У даному випадку задача мінімізації (2.1) набуває вигляду:

$$S = S(\beta_0, \beta_1) = \sum_{i=1}^{n} \left[y_i - (\beta_0 + \beta_1 x_i)\right]^2 \to min, \qquad (2.4)$$

де мінімум береться по всім значенням β_0, β_1 при фіксованих $x_1,\ldots,x_n$ та $y_1,\ldots, y_n$. Позначимо розв'язок задачі (2.4) (b_0, b_1), а відповідну оцінку функції регресії (2.3) – $\hat{y}$, тобто $\hat{y} = \hat{y}(x) = b_0 + b_1 x$.

На наступному малюнку схематично зображена вказана пряма регресії і набір експериментальних точок (x_i, y_i). Окрім цього зображені вертикальні відрізки, що з'єднують зазначені точки і пряму.

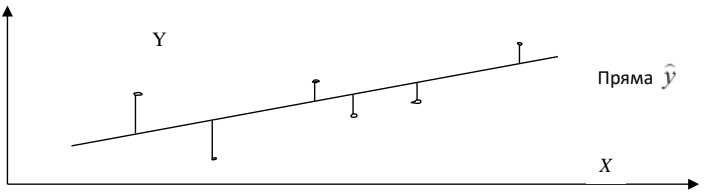


Рис. 2.2.1. Вертикальні відхилення, що мінімізують суму квадратів в методі найменших квадратів.

Легко зрозуміти, що при будь-якому способі проведення прямої лінії для апроксимації залежності між y та x будуть мати місце відхилення експериментальних точок від обраної прямої (якщо тільки всі ці точки не лежать на одній прямій, але останнє практично ніколи не трапляється). Ці відхилення зручно вимірювати різницями ординат, що відповідають експериментальним точкам та точкам апроксимуючої прямої при значеннях $x = x_1, \ldots, x_n$, тобто алгебраїчними величинами вертикальних відрізків типу зображених на мал. 2.1. Якщо апроксимація виконується згідно з МНК, то сума квадратів довжин таких відрізків буде найменшою з можливих.

Зрозуміло, що β_1 та β_0 є, відповідно, кутовим коефіцієнтом та вільним членом (відрізком на осі ординат при $x = 0$) прямої (2.2), а b_1 та b_0 – їх оцінками за експериментальними даними. При цьому останні є, відповідно, кутовим коефіцієнтом та вільним членом рівняння прямої (2.10) ((2.11)).

Система нормальних рівнянь. Дослідження та розв'язок задачі (2.4).

Для розв'язання задачі (2.4) обчислимо частинні похідні по β_0, β_1 функції $S = S(\beta_0, \beta_1)$. Маємо

$$\partial S / \partial \beta_0 = - 2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i),$$

$$\partial S / \partial \beta_1 = - 2 \sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i).$$

Прирівнюючи знайдені похідні нулю і виконуючи належні спрощення (рекомендується виконати їх самостійно), приходимо до системи двох рівнянь, в якій, підкреслимо, невідомими є параметри β_0, β_1:

$$\beta_0 n + \beta_1 \Sigma x_i = \Sigma y_i,$$

$$\beta_0 \Sigma x_i + \beta_1 \Sigma x_i^2 = \Sigma x_i y_i, \tag{2.5}$$

де для спрощення запису індекси підсумовування опущено. Ця система носить назву *системи нормальних рівнянь* або *нормальної системи*. В одному з наступних розділів буде доведено, що розв'язок нормальної системи і справді є розв'язком задачі (2.4).

Розглянемо спочатку випадок існування та єдиності розв'язку системи (2.5). Тобто ми поки що вважаємо визначник даної системи відмінним від 0. Позначимо шуканий розв'язок (b_0, b_1); відносно β_1 маємо (переконайтесь у цьому):

$$b_1 = \frac{\sum x_i y_i - [(\sum x_i)(\sum y_i)]/n}{\sum x_i^2 - (\sum x_i)^2/n}, \tag{2.6}$$

де підсумовування ведеться по i від 1 до n. Розв'язок нормальної системи відносно β_0 має такий вигляд:

$$b_0 = \bar{y} - b_1 \bar{x}, \tag{2.7}$$

де $\bar{y}$ та $\bar{x}$ – відповідні середні арифметичні значення:

$$\bar{y} = (y_1 + \ldots + y_n)/n, \ \bar{x} = (x_1 + \ldots + x_n)/n.$$

Зауваження. Оцінка b_1 параметра β_1 може бути записаною також іншим чином:

$$b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}. \tag{2.8}$$

Дійсно, це твердження випливає з наступних рівностей

$\Sigma(x_i - \bar{x})(y_i - \bar{y}) = \Sigma x_i y_i - \bar{x}\Sigma y_i - \bar{y}\Sigma x_i + n\bar{x}\bar{y} = \Sigma x_i y_i - n\bar{x}\bar{y} =$

$= \Sigma x_i y_i - (\Sigma x_i)(\Sigma y_i)/ n.$

Зручно ввести позначення:

$$S_{xy} = \Sigma(x_i - \bar{x})(y_i - \bar{y}),\ S_{xx} = \Sigma(x_i - \bar{x})^2,\ S_{yy} = \Sigma(y_i - \bar{y})^2.$$

Тоді маємо легкий для запам'ятовування вираз оцінки β_1:

$$b_1 = S_{xy} / S_{xx}. \tag{2.9}$$

Вище була введена функція

$$\hat{y} = \hat{y}(x) = b_0 + b_1 x \tag{2.10}$$

Величина $\hat{y} = \hat{y}(x)$ *є оцінкою функції регресії* $g(x)$ з рівності (2.2). Іноді зручніше використовувати дещо інший вираз для $\hat{y}(x)$:

$$\hat{y}(x) = \bar{y} + b_1(x - \bar{x}). \tag{2.11}$$

Ця рівність одержується з (2.10) підстановкою замість b_0 його виразу (2.7).

З рівності (2.11) одразу бачимо, що пряма лінія, яка зображує у декартовій площині залежність $\hat{y} = \hat{y}(x)$, проходить через точку декартової площини з координатами $(\bar{x}, \bar{y})$.

Припустимо тепер, що визначник Δ системи (2.5) дорівнює 0. З'ясуємо, за яких умов це трапляється. Маємо

$$\Delta = n\,\Sigma x_i^2 - (\Sigma x_i)^2.$$

Оскільки $(\Sigma x_i)^2 = (\Sigma 1 \cdot x_i)^2$, то за нерівністю Коші – Буняковського буде $(\Sigma x_i)^2 \le \Sigma 1^2 \cdot \Sigma x_i^2 = n\,\Sigma x_i^2$, отже завжди

$$\Delta \ge 0 \tag{2.12}$$

Скористаємось відомою умовою наявності рівності в нерівності Коші – Буняковського. Бачимо, що коли ввести n-вимірні вектори $\mathbf{1} = (1,\ldots,1)$ і $\boldsymbol{x} = (x_1,\ldots, x_n)$, то рівність в (2.12) буде мати місце тоді і тільки тоді, коли вектори $\mathbf{1}$ і $\boldsymbol{x}$ будуть лінійно залежними. Це означає, що знайдуться такі сталі c_1, c_2, що одночасно не дорівнюють 0 і для яких виконується рівність

$$c_1 \cdot \mathbf{1} + c_2 \cdot \boldsymbol{x} = 0, \tag{2.13}$$

де права частина – це нульовий вектор $(0,\ldots,0)$. При цьому випадок $c_2 = 0$, очевидно, не може мати місця (бо інакше, оскільки тоді $c_1 \neq 0$, буде $\mathbf{1} = 0$). Тому з (2.13) випливає, що $\boldsymbol{x} = (c,\ldots,c)$, де c – деяка стала. Це означає, що всі спостереження робляться в одній точці: $x_1 = \ldots = x_n = c$. За вказаною умовою система (2.5) зводиться, фактично, до одного-єдиного рівняння відносно β_0, β_1:

$$\beta_0 n + \beta_1 nc = \Sigma\, y_i, \tag{2.14}$$

яке має безліч дійсних розв'язків. Відповідно маємо нескінченну кількість рівнянь регресії, яким геометрично відповідає сукупність всіх прямих, що проходять через точку $(c,\bar{y})$

Висновок. З розглянутого вище бачимо, зокрема, що нормальна система рівнянь (2.5) завжди сумісна, незалежно від того, дорівнює її детермінант 0 чи не дорівнює. В останньому випадку маємо єдиний розв'язок, що дається рівностями (2.6) ((2.8)) і (2.7). Перший випадок може трапитись тільки тоді, коли всі спостереження проводяться лише при одному значенні x. У цьому випадку вказана система має безліч розв'язків, кожен з яких може бути знайдений з рівняння (2.14).

2.2.3 Властивості оцінок простої лінійної регресії

2.2.3.1. Вектор залишків простої лінійної регресії та деякі його властивості.

Визначимо так звані *залишки* регресійної моделі (2.3). За означенням, це величини $e_i = y_i - \hat{y}_i$, де $\hat{y}_i = \hat{y}(x_i)$, $i = 1,\ldots, n$. *Вектором залишків* моделі (2.3) називається вектор $e = (e_1,\ldots, e_n)$. Зауважимо, що коли з самого початку вільний член β_0 входив у модель (тобто не покладався одразу рівним 0), то сума всіх залишків (інакше, сума координат вектору залишків) дорівнює 0, тобто

$$\sum_{i=1}^{n} e_i = 0. \tag{3.1}$$

Це одразу випливає з рівності $\partial S/\partial\beta_0 = 0$ при $\beta_0 = b_0$, $\beta_1 = b_1$ (див. співвідношення після системи (2.4)). Зазначена властивість використовується, наприклад, при перевірці обчислень за МНК, якщо останні виконувалися вручну.

Зауваження. Корисно помітити, що середнє арифметичне значень $\hat{y}(x_i)$, $i = 1, 2, \ldots, n$ дорівнює $\bar{y}$ – середньому значенню спостережуваних відгуків $y_1,\ldots,y_n$. Тобто, коли позначити

$$\hat{y}_i = \hat{y}(x_i),\ i = 1, 2, \ldots, n;\ \bar{\hat{y}} = (\hat{y}_1 + \ldots + \hat{y}_n)/n,$$

то матимемо рівність

$$\bar{\hat{y}} = \bar{y}. \tag{3.2}$$

Дійсно, згідно з рівністю (2.10) маємо

$$\bar{\hat{y}} = \sum_{i=1}^{n} \hat{y}_i/n = \left(\sum_{i=1}^{n} (\bar{y} + b_1(x_i - \bar{x}))\right)/n = \bar{y} + b_1\bar{x} - b_1\bar{x} = \bar{y}.$$

З рівності (3.1) одразу випливає вже відзначений вище факт рівності 0 суми координат вектора залишків *e*:

$$\Sigma\, e_i = \Sigma\,(y_i - \hat{y}_i) = n\ \bar{y} - n\,\bar{\hat{y}} = 0.$$

(Нагадаємо, що коли не робиться спеціальних роз'яснень, то за відсутністю індексів у символі Σ мається на увазі підсумовування від 1 до *n*).

2.2.3.2. Про одну властивість оцінок МНК

У багатьох питаннях регресійного аналізу є корисною наступна рівність

$$\Sigma(y_i - \bar{y})^2 = \Sigma(y_i - \hat{y}_i)^2 + \Sigma(\hat{y}_i - \bar{y})^2 . \qquad (3.3)$$

Дана рівність часто називається *основною тотожністю дисперсійного аналізу*. (Зміст цієї назви стане зрозумілим дещо пізніше.) Сама ж рівність (3.2) може бути одержана наступним чином.

$$\Sigma(y_i - \bar{y})^2 = \Sigma(y_i - \hat{y}_i + \hat{y}_i - \bar{y})^2 = \Sigma(y_i - \hat{y}_i)^2 + \Sigma(\hat{y}_i - \bar{y})^2 +$$

$$+ 2\Sigma(\hat{y}_i - \bar{y})\,(\hat{y}_i - \bar{y}).$$

Тепер досить довести, що остання сума дорівнює 0. Використовуючи рівність (2.10), маємо

$$\hat{y}_i - \bar{y} = b_1(x_i - \bar{x}),\ y_i - \hat{y}_i = y_i - \bar{y} - b_1(x_i - \bar{x}).$$

Звідси вказана сума дорівнює

$$\Sigma\, b_1(x_i - \bar{x})((y_i - \bar{y}) - b_1(x_i - \bar{x})) = b_1\,(S_{xy} - b_1\,S_{xx}) = 0$$

(врахувати рівність (2.8)). Рівність (3.2) доведено.

З аналогічних міркувань також зрозуміло, що

$$\Sigma(\hat{y}_i - \bar{y})^2 = \Sigma(b_1(x_i - \bar{x}))^2 = b_1^2 S_{xx} = b_1 S_{xy}. \quad (3.4)$$

Зауважимо, що суми квадратів у рівності (3.3) мають спеціальні назви. Сума зліва – сума квадратів відносно середнього; перша сума справа – сума квадратів відносно регресії; друга сума справа – сума квадратів, що зумовлена регресією.

2.2.3.3. Пояснювана частина варіації даних

Позначимо

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} . \quad (3.5)$$

З рівності (3.3) одразу випливає нерівність

$$R^2 \le 1. \quad (3.6)$$

Можна вважати, що величина R^2 вимірює „долю загального розкидання даних, що пояснюється регресією". Її часто вимірюють в процентах, помножуючи на 100. Досить часто величина R^2 носить назву „***коефіцієнт детермінації***". Величина R^2 виводиться на друк у більшості відомих комп'ютерних програм з регресійного аналізу. Чим ближчою є величина R^2 до 1, тим краще функція регресії (2.9) відповідає дійсному характеру зв'язку між незалежною та залежною змінними.

2.2.3.4. Зв'язок величини R^2 з вибірковими коефіцієнтами кореляції R_{xy} та $R_{y\hat{y}}$.

Як відомо, коефіцієнтом кореляції між випадковими величинами ξ, η називається вираз

$$\rho_{\xi\eta} = \mathrm{Cov}(\xi,\ \eta) / (D\xi \cdot D\eta)^{1/2},$$

де $\mathrm{Cov}(\xi,\ \eta) = M\xi\eta - M\xi\, M\eta$, D – символ дисперсії.

Оцінкою коефіцієнта кореляції (або *вибірковим* коефіцієнтом кореляції) між двома величинами ξ та η є вираз

$$R_{\xi\eta} = \frac{\sum(\xi_i - \overline{\xi})(\eta_i - \overline{\eta})}{\left(\sum(\xi_i - \overline{\xi})^2 \sum(\eta_i - \overline{\eta})^2\right)^{1/2}}\ , \tag{3.7}$$

де $(\xi_i,\ \eta_i)$, $i = 1,..., n$ – значення (ξ,η) в n незалежних експериментах, $\overline{\xi}$ та $\overline{\eta}$ – відповідні середні арифметичні, а підсумовування виконується від 1 до n.

Позначимо R_{xy} та $R_{y\hat{y}}$, відповідно, вибіркові коефіцієнти кореляції між x та y і y та $\hat{y}$ відповідно. Тоді мають місце рівності

$$R_{y\hat{y}} = sign\ (b_1) \cdot R_{xy} \tag{3.8}$$

де

$$sign\ x = \begin{cases} 1, & x > 0; \\ 0, & x = 0; \\ -1, & x < 0. \end{cases}$$

$$R^2 = (R_{xy})^2, \tag{3.9}$$

$$R^2 = (R_{y\hat{y}})^2 \tag{3.10}$$

Дійсно, з використанням (3.4) одержуємо

$$R_{y\hat{y}} = \frac{\sum(y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\left(\sum(y_i - \overline{y})^2 \sum(\hat{y}_i - \overline{\hat{y}})^2\right)^{1/2}} = \frac{b_1 \sum(y_i - \overline{y})(x_i - \overline{x})}{|b_1|\left(\sum(y_i - \overline{y})^2 \sum(x_i - \overline{x})^2\right)^{1/2}} = sign(b_1) \cdot R_{xy} \tag{3.11}$$

З іншого боку,

$$R^2 = \frac{b_1^2 \sum (x_i - \bar{x})^2}{\sum (y_i - \bar{y})^2} = \frac{S_{xy}^2}{S_{xx} S_{yy}} = (R_{xy})^2 .$$

(3.11) і останні співвідношення доводять рівності (3.8) — (3.10).

2.2.4. Ймовірнісні припущення про випадкову складову моделі простої лінійної регресії та їх наслідки

2.2.4.1. Незалежність, однорідність і відсутність систематичних похибок.

Надалі буде вважатися, що всі експерименти є незалежними, виконуються в однакових умовах і не мають систематичних похибок. Математично це виражається наступним чином. Нехай ε_i позначає величину похибки в i-му експерименті (тобто $\varepsilon_i = y_i - (\beta_0 + \beta_1 x_i)$), $i = 1,..., n$ Тоді вектор похибок $\varepsilon = (\varepsilon_1,..., \varepsilon_n)$ становить собою сукупність незалежних однаково розподілених випадкових величин, причому математичні сподівання кожної з цих величин дорівнюють 0:

$$M\varepsilon_i = 0,\ i = 1,..., n, \tag{4.1}$$

а дисперсії дорівнюють деякій сталій σ^2:

$$D\varepsilon_i = \sigma^2,\ i = 1,..., n. \tag{4.2}$$

2.2.4.1.1. Зауваження. З (4.1) та (4.2) одразу випливають рівності (переконайтеся в цьому):

$$My(x) = \beta_0 + \beta_1 x, \tag{4.3}$$

$$D\, y(x) = \sigma^2 , \tag{4.4}$$

$$D\, \bar{y} = \sigma^2 / n \tag{4.5}$$

2.2.4.1.2. Наслідки. Наслідками зроблених вище припущень є також наступні властивості оцінок параметрів моделі:

$$1)\ Mb_0 = \beta_0,\quad Mb_1 = \beta_1; \tag{4.6}$$

$$2)\ Db_0 = \frac{\sum x_i^2}{n\sum (x_i - \bar{x})^2}\sigma^2,\quad Db_1 = \frac{\sigma^2}{\sum (x_i - \bar{x})^2}. \tag{4.7}$$

$$3)\ Cov(\bar{y}, b_1) = 0, \tag{4.8}$$

$$4)\ Cov(b_0,b_1) = -\bar{x}\frac{\sigma^2}{\sum (x_i - \bar{x})^2}\ . \tag{4.9}$$

5) Нехай x_0 – довільне значення змінної x. Позначимо $\hat{y}_0$ значення оцінки функції регресії $\hat{y}$ в точці x_0. Тоді має місце рівність

$$D\,\hat{y}_0 = \sigma^2\left[\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right]. \tag{4.10}$$

Зокрема, рівності (4.6) означають, що b_0 та b_1 є незсуненими оцінками, відповідно, величин β_0 та β_1. Рівності (4.7) дають вирази дисперсій оцінок коефіцієнтів регресії через дисперсію випадкової складової моделі (2.3). Рівність (4.8) стверджує некорельованість величин $\bar{y}$ та b_1. Рівність (4.9) дає явний вираз коваріацій між оцінками b_0,b_1, а (4.10) — вираз дисперсії оцінки функції регресії у довільній точці спостережень. З останньої рівності одразу бачимо, що дисперсія величини $\hat{y}_0$ є мінімальною, коли точка x_0 співпадає з $\bar{x}$ і зростає при віддаленні цієї точки від $\bar{x}$.

2.2.**4.1.3. Доведення властивостей 1) — 5)**

Доведемо спочатку властивість 1). Маємо $Mb_1 = M\dfrac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sum(x_i-\bar{x})^2}$. При цьому $\sum(x_i-\bar{x})(y_i-\bar{y}) = \sum(x_i-\bar{x})y_i$ (Переконайтеся у вірності останньої рівності). Враховуючи ще рівність

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{4.11}$$

одержуємо

$$Mb_1 = \frac{\sum(x_i-\bar{x})My_i}{\sum(x_i-\bar{x})^2} = \frac{\sum(x_i-\bar{x})(\beta_0+\beta_1 x_i)}{\sum(x_i-\bar{x})^2} = \beta_1\frac{\sum(x_i-\bar{x})x_i}{\sum(x_i-\bar{x})^2} = \beta_1\frac{\sum(x_i-\bar{x})(x_i-\bar{x})}{\sum(x_i-\bar{x})^2} =$$

$$= \beta_1\frac{\sum(x_i-\bar{x})^2}{\sum(x_i-\bar{x})^2} = \beta_1.$$

$$Mb_0 = M\bar{y} - Mb_1\bar{x} = \beta_0 + \beta_1\bar{x} - \beta_1\bar{x} = \beta_0.$$

Рівності (4.6) доведено.

Аналогічним чином отримуємо:

$$Db_1 = \frac{\sum(x_i-\bar{x})^2 Dy_i}{\left(\sum(x_i-\bar{x})^2\right)^2} = \frac{\sum(x_i-\bar{x})^2\sigma^2}{\left(\sum(x_i-\bar{x})^2\right)^2} = \frac{\sigma^2}{\sum(x_i-\bar{x})^2}.$$

Другу з нерівностей (4.7) доведено. Першу з зазначених нерівностей буде доведено трохи пізніше. Перед доказом властивості 3) нагадаємо, що для випадкових величин ξ,η їх коваріація $Cov(\xi,\eta)$ обчислюється за формулою $Cov(\xi,\eta) = M\xi\eta - M\xi\, M\eta$. Згадаємо також кілька властивостей коваріацій випадкових величин та коваріаційних матриць випадкових векторів.

1. Для випадкової величини ξ має місце рівність $Cov(\xi,\xi) = D\xi$.

2. Для випадкових величин ξ, η, ξ_1, ξ_2, η_1, η_2 та сталих a_1, a_2 мають місце рівності

$$Cov(a_1\xi_1 + a_2\xi_2, \eta) = a_1 Cov(\xi_1, \eta) + a_2\, Cov(\xi_2, \eta),$$

$$Cov(\xi, a_1\eta_1 + a_2\eta_2) = a_1 Cov(\xi, \eta_1) + a_2\, Cov(\xi, \eta_2).$$

3. Нехай $D(\xi,\eta)$ – коваріаційна матриця випадкових **векторів** ξ, η , що мають компоненти ξ_1, ξ_2,… та η_1, η_2,… відповідно (так що на місті (i,j) матриці $D(\xi, \eta)$ стоїть величина $\mathrm{Cov}(\xi_i, \eta_j)$); нехай також A, B – невипадкові матриці. Тоді $D(A\xi, B\eta) = AD(\xi,\eta)B'$, де B' – транспонована до B матриця.

Повернемося до доказу властивості 3) оцінок МНК. Нехай $Y = (y_1, y_2,...,y_n)'$ – n-вимірний вектор-стовбець спостережень, $A = \left(\frac{1}{n}, \frac{1}{n},..., \frac{1}{n}\right)$, $B = \left(\frac{x_1 - \bar{x}}{S_{xx}}, \frac{x_2 - \bar{x}}{S_{xx}},..., \frac{x_n - \bar{x}}{S_{xx}}\right)$. Легко бачити, що $\bar{y} = AY$, $b_1 = BY$. При цьому $DY = \sigma^2 I$, де I – одинична матриця розміру $n \times n$. Отже, використовуючи властивість 3. коваріаційної матриці, одержуємо

$$Cov(\bar{y}, b_1) = Cov(AY, BY) = A(DY)B' = \frac{\sigma^2}{nS_{xx}} \sum_{i=1}^{n} (x_i - \bar{x}) = 0,$$

тобто рівність (4.8) доведено. Тепер, користуючись відомою властивістю дисперсії суми некорельованих випадкових величин, одержимо:

$$Db_0 = D(\bar{y} - b_1\bar{x}) = D\bar{y} + \bar{x}^2 Db_1 = \frac{\sigma^2}{n} + \bar{x}^2 \frac{\sigma^2}{\sum (x_i - \bar{x})^2} = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2}\right] =$$

$= \frac{\sigma^2 \sum x_i^2}{n \sum (x_i - \bar{x})^2}$, що й доводить першу з нерівностей (4.7). Для доказу (4.9), використовуючи відомі властивості коваріацій випадкових величин, знайдемо:

$Cov(b_0,b_1) = Cov(\bar{y} - b_1 \bar{x}, b_1) = Cov(\bar{y}, b_1) + Cov(-b_1 \bar{x}, b_1) = -\bar{x}\, Cov(b_1, b_1) =$

$= -\bar{x}\, Db_1$.

Тепер лишається скористатися другою з властивостей 2).

Нарешті, згідно з рівностями (2.10) та (4.8), маємо

$$D\hat{y}_0 = D[\bar{y} + (x_0 - x) b_1] = D\bar{y} + (x_0 - x)^2 D b_1.$$

Тепер лишається скористатися рівностями (4.5) та (4.7). Зауважимо, що дисперсія величини $\hat{y}_0$ є мінімальною, коли точка x_0 співпадає з $\bar{x}$ і зростає при віддаленні цієї точки від $\bar{x}$.

2.2.4.2. Нормальність випадкової складової. Надалі постійно припускатиметься, що випадкова складова ε регресійної моделі (2.3) є нормально розподіленою випадковою величиною з нульовим математичним сподіванням і деякою (невідомою) дисперсією σ^2:

$$\varepsilon \in N(0, \sigma). \tag{4.12}$$

Припущення, зроблені вище, зумовлюють, що вектор випадкових складових $E = (\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_n)$, де $\varepsilon_i = y_i - (\beta_0 + \beta_1 x_i)$, $i = 1,2,\ldots,n$, є нормально розподіленим випадковим вектором з взаємно незалежними компонентами, кожна з яких має тип (4.12). Зокрема, коваріаційна матриця $D(E)$ вектора E є діагональною з елементами σ^2 на головній діагоналі:

$$D(E) = \begin{bmatrix} \sigma^2 & 0 & ... & 0 & 0 \\ 0 & \sigma^2 & ... & 0 & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & \sigma^2 & 0 \\ 0 & 0 & ... & 0 & \sigma^2 \end{bmatrix} = \sigma^2 I_n, \quad (4.13)$$

де I_n – одинична матриця розміру $n \times n$.

Позначимо ще Y вектор спостережень, тобто $Y = (y_1,\ldots, y_n)$. Зауважимо, що оскільки $\beta_0 + \beta_1 x_i$ — є невипадковими величинами, то

$$D(Y) = D(E). \quad (4.14)$$

З припущення (4.12) випливає дуже багато корисних наслідків. Зокрема, такими є дві наступні теореми, справедливість яких буде наведено у подальших розділах курсу як наслідок деяких більш загальних положень.

2.4.2.1. Теорема. Остаточна сума квадратів, поділена на σ^2, має розподіл χ^2 з $n - 2$ степенями волі, коротше:

$$RSS / \sigma^2 \in \chi^2_{n-2},$$

де $RSS = \Sigma(y_i - \hat{y}_i)^2$.

. Зауваження. Нехай відомо, що відношення деякої випадкової величини ξ до деякої сталої величини (параметру) θ має розподіл χ^2 з m степенями волі. Тоді відношення ξ / m є незсуненою оцінкою для θ.

□ Дійсно, оскільки ξ / θ має розподіл χ^2_m, то $M(\xi / \theta) = m$, тому $M(\xi / m) = \theta$. □

Позначимо

$$RSS/(n - 2) = S^2 . \quad (4.15)$$

2.2.4.2.3. Наслідок. Величина S^2 є незсуненою оцінкою величини σ^2:

$$M\,S^2 = \sigma^2 \tag{4.16}$$

2.2.4.2.4. Зауваження. Можна довести, що рівність (4.16) є справедливою і без припущення щодо нормальності розподілу ξ (за виконанням інших зроблених вище припущень щодо випадкової складової моделі (2.3).

2.2.4.2.5. Наслідок. Величини

$$\hat{D}\,b_0 = \frac{\sum x_i^2}{n\sum (x_i - \bar{x})^2} S^2, \quad \hat{D}\,b_1 = \frac{1}{\sum (x_i - \bar{x})^2}\ S^2 \tag{4.17}$$

є незсуненими оцінками величин Db_0 та Db_1 відповідно.

2.2.4.2.6. Теорема. За умови (4.12) мають місце співвідношення

$$\frac{b_0 - \beta_0}{\sqrt{\hat{D}b_0}} \in t_{n-2}, \tag{4.18}$$

$$\frac{b_1 - \beta_1}{\sqrt{\hat{D}b_1}} \in t_{n-2}\ , \tag{4.19}$$

де t_{n-2} — розподіл Стьюдента з $n - 2$ степенями волі.

2.2.4.2.7. Наслідок. Нехай $u_{1-\alpha/2}^{t_{n-2}}$ – квантиль рівня $1 - \alpha/2$ розподілу Стьюдента t_{n-2}. Тоді інтервали

$$B_0 = [(b_0 - u_{1-\alpha/2}^{t_{n-2}}\sqrt{\hat{D}b_0}, \quad b_0 + u_{1-\alpha/2}^{t_{n-2}}\sqrt{\hat{D}b_0}\ ],$$

$$B_1 = [(b_1 - u_{1-\alpha/2}^{t_{n-2}}\sqrt{\hat{D}b_1}, \quad b_1 + u_{1-\alpha/2}^{t_{n-2}}\sqrt{\hat{D}b_1}\ ],$$

є довірчими інтервалами рівня α для параметрів b_0 та b_1 відповідно.

□ Це випливає із співвідношень (4.18), (4.19) та тією властивістю квантилів неперервних випадкових величин, що $F(u_p) = p$, де F – функція розподілу даної випадкової величини, u_p – її квантиль рівня p. □

2.2.4.2.8. Зауваження про перевірку гіпотез щодо значень коефіцієнтів β_0, β_1.

Щойно наведені результати дають змогу перевіряти гіпотези $H_0^{\beta_0}: \beta_0 = b_0{}^0$ та $H_0^{\beta_1}: \beta_1 = b_1{}^0$, де $b_0{}^0$ та $b_1{}^0$ – фіксовані числа. Ідея перевірки є дуже простою. Якщо число $b_0{}^0$ належить інтервалові B_0, то гіпотеза $H_0^{\beta_0}$ приймається, у протилежному випадку – не приймається. Аналогічно перевіряється гіпотеза $H_0^{\beta_1}$.

Досить часто, зокрема, в комп'ютерних реалізаціях даної процедури перевірки використовуються дещо інші (формально, але не по суті) дії. А саме, позначимо $T(b_0)$ ($T(b_1)$) величину з правої частини (4.18) ((4.19)) при $\beta_0 = b_0{}^0$ ($\beta_1 = b_1{}^0$). Ця величина порівнюється з $u_{1-\alpha/2}^{t_{n-2}}$ – квантилем рівня $1 - \alpha/2$ розподілу Стьюдента t_{n-2}. Відповідна гіпотеза не приймається, якщо за абсолютною величиною вказана величина перевищує даний квантиль. Іншими словами, гіпотеза не приймається, якщо $T(b_0)$ (відповідно, $T(b_1)$) попадає у двосторонню критичну множину $(-\infty, -u_{1-\alpha/2}^{t_{n-2}}) \cup (u_{1-\alpha/2}^{t_{n-2}}, +\infty)$. За наявністю альтернативної гіпотези $\{b_i \geq b_i{}^0\}$ ($\{b_i \leq b_i{}^0\}$), $i \in \{0,1\}$, доцільніше використовувати односторонню критичну множину $(u_{1-\alpha}^{t_{n-2}}, +\infty)$ ($(-\infty, -u_{1-\alpha}^{t_{n-2}})$

Особливо часто доводиться перевіряти гіпотезу $H_0^{\beta_1}$ при $b_1{}^0 = 0$ (тобто мова йде про перевірку гіпотези $\{\beta_1 = 0\}$. У цьому випадку $H_0^{\beta_1}$ називається *гіпотезою про незначимість коефіцієнту регресії*. Якщо вона приймається, то можна вважати, що *y* не залежить від *x* (в рамках лінійної моделі(2.3)).

2.2.4.2.9. Зауваження. Деякі поширені комп'ютерні статистичні програми, наприклад, Statgraphics 3.0, використовують дещо іншу техніку перевірки гіпотез. А саме, не фіксуються заздалегідь критичні множини, а обчислюються так звані *P*-значення (*P*-values). Зокрема, при перевірці гіпотези $H_0^{\beta_0}$ ($H_0^{\beta_1}$) *P*-значення є ймовірністю того, що за умови справедливості даної гіпотези, статистика $T(b_0)$ (відповідно, $T(b_1)$) буде за абсолютною величиною рівною або більшою того значення, що конкретно отримано. Малість *P*-значення свідчить про недоцільність довіри до цієї гіпотези. Навпаки, великі *P*-значення свідчать на користь гіпотез, що перевіряються. Так, малі *P*-значення при перевірці гіпотези $H_0^{\beta_1}$ з $b_1{}^0 = 0$ свідчать *про значимість* коефіцієнту регресії β_1. З коментарів, які містяться у відповідних роздруківках, можна зрозуміти, що вважається малим, а що – великим *P*-значенням у кожному конкретному випадку.

2.2.4.3. Перевірка адекватності моделі (2.3).

Наведені вище результати мають місце, якщо модель (2.3) є адекватною, тобто якщо спостережувана величина *y* насправді задовольняє рівності (2.3). Вичерпним чином перевірити виконання цього у більшості випадків неможливо. Проте цілком можливо перевірити, відповідають чи ні, висновки, що зроблені на основі припущення про виконання (2.3), поведінці

реальних даних. Можуть досліджуватися різні аспекти такої відповідності. Кожне таке дослідження можна називати перевіркою моделі на адекватність. У зазначеному напрямку часто перевіряється достатня близькість оцінок дисперсії випадкової складової спостережень, що одержано, з одного боку, з застосуванням моделі (2.3), а з іншого — незалежним від постульованої моделі способом.

Розглянемо вказаний метод перевірки адекватності більш детально. Одразу зауважимо, що *для його реалізації хоча б одній з точок $\{x_i\}$ потрібно мати більш ніж одне спостереження.*

Нехай x_1, x_2,…, x_n — точки спостережень, $n > 1$. Всі ці точки вважаються різними. Припустимо, що при $x = x_i$ спостерігалися значення y_{i1}, y_{i2},…, $y_{i m_i}$. Розглянемо середні арифметичні залежної змінної в кожній точці спостережень, тобто величини

$$\bar{y}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} y_{ij}, \; i = 1,2,\ldots,n. \tag{4.20}$$

Визначимо також величини

$$s_i^2 = \frac{1}{m_i - 1} \sum_{j=1}^{m_i} (y_{ij} - \bar{y}_i)^2, \; i = 1,2,\ldots,n. \tag{4.21}$$

Якщо дисперсії спостережень однакові у всіх точках (дорівнюють σ^2), то як відомо з попереднього (теорема про $\bar{x}$, s^2 при теоретичному нормальному розподілі), має місце співвідношення

$$\frac{m_i - 1}{\sigma^2} s_i^2 \in \chi^2(m_i - 1), \tag{4.22}$$

(тобто ліва частина (4.22) розподілена за законом χ^2-квадрат з $m_i - 1$ степенями волі.) Звідси випливає, що

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}(y_{ij}-\bar{y}_i)^2 / \sigma^2 \in \chi^2(N-n), \tag{4.23}$$

де $N = m_1 + \ldots + m_n$. Таким чином (врахувати зауваження 4.2.2), величина

$$S^2{}_2 = \frac{1}{N-n}\sum_{i=1}^{n}\sum_{j=1}^{m_i}(y_{ij}-\bar{y}_{i\cdot})^2 \tag{4.24}$$

є незсуненою оцінкою величини σ^2, причому цей факт не залежить від справедливості чи несправедливості гіпотези про адекватність моделі (2.3).

Покладемо

$$S^2{}_1 = \frac{1}{n-2}\sum_{i=1}^{n} m_i(\bar{y}_{i\cdot}-\hat{y}_i)^2, \tag{4.25}$$

2.2.**4.3.1. Твердження**. В разі виконання рівності (2.3) величини $S^2{}_1$ та $S^2{}_2$ є незалежними одна від одної, причому обидві вони є незсуненими оцінками величини σ^2.

Дійсно, у будь-якому разі має місце рівність (див. вправу 4.3.2)

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}(y_{ij}-\hat{y}_i)^2 = \sum_{i=1}^{n}\sum_{j=1}^{m_i}(y_{ij}-\bar{y}_i)^2 + \sum_{i=1}^{n} m_i(\hat{y}_i-\bar{y}_i)^2. \tag{4.26}$$

З теореми 4.2.1 випливає, що величина у лівій частині (4.26), поділена на σ^2, має розподіл χ^2 з $N - 2$ степенями волі, де $N = m_1 + \ldots + m_n$. Позначимо першу і другу суму з правої частини (4.26) через Z_1 та Z_2 відповідно. Щойно доведено, що Z_1/σ^2 має розподіл χ^2 з $N - n$ степенями волі. Якщо тепер скористатися наведеною нижче теоремою 4.3.3, то одержимо, що Z_2/σ^2 має

розподіл χ^2 з $N - 2 - (N - n) = n - 2$ степенями волі, причому величини Z_1/σ^2 і Z_2/σ^2 не залежать одна від одної. З першого положення випливає, що величина $S^2_1 = Z_1/(n - 2)$ є незсуненою оцінкою σ^2. З другого положення випливає, що величини $S^2_1 = Z_1\sigma^2/(\sigma^2(n - 2))$ та $S^2_2 = Z_2\sigma^2/(\sigma^2(N - n))$є незалежними.

2.2.4.3.2. Вправа. Доведіть рівність (4.26) самостійно, скориставшись міркуваннями типу тих, що використовувалися при доведенні рівності (3.2).

2.2.4.3.3. Теорема. Нехай $\xi = (\xi_1,\ldots, \xi_n)$ — випадковий вектор з нормальним розподілом $N(\mu, I_n)$, $\mu = (\mu_1,\ldots,\mu_n)$, Q_i, $i = 1,2$ — квадратичні форми від $\xi_1 - \mu_1,\ldots, \xi_n - \mu_n$. Тоді якщо $Q_i \in \chi^2(r_i)$, причому $Q_1 - Q_2 \geq 0$, то $Q_1 - Q_2$ та Q_2 є незалежними і мають розподіли $\chi^2(r_1 - r_2)$ та $\chi^2(r_2)$ відповідно.

2.4.3.4. Статистика для перевірки гіпотези про адекватність.

Із встановленого вище випливає співвідношення

$$S^2_1 / S^2_2 \in F(n - 2, N - n), \tag{4.27}$$

де F — розподіл Фішера. Якщо розглядувана модель є адекватною, величини S^2_1 та S^2_2 оцінюють однаковий параметр σ^2. Звідси випливає процедура перевірки досліджуваної умови, що використовує вираз з лівої частини (4.27) (F-статистику) в якості статистики критерію і з квантилями розподілу $F(n - 2, N - n)$ в якості критичних точок.

2.2.4.3.5. Зауваження. Результати перевірки гіпотези про незначимість регресії за методикою п. 4.2.8 стають більш вагомими, якщо є підтверджуючі результати перевірки гіпотези про адекватність моделі.

2.2.4.3.6. Зауваження. Згадану гіпотезу про незначимість коефіцієнту регресії (яка полягає в тому, що $\beta_1 = 0$) можна також перевірити, вживаючи F-розподіл Фішера. Наприклад, це випливає з того, що коли деяка випадкова величина має розподіл Стьюдента t_n, то квадрат цієї величини має розподіл Фішера $F_{1,n}$. У поширених комп'ютерних програмах перевірка зазначеної гіпотези виконується з використанням обох розподілів. Для випадку розглядуваної тут простої лінійної регресії статистика критерію, що використовує розподіл $F_{1,n}$ є просто квадратом статистики $T(b_1)$ при $b_1^0 = 0$ (див. нижче зауваження 4.3.8). Звідси, по-перше, стає очевидним, що при використанні $F_{1,n}$ для всіх трьох основних альтернатив ($\beta_1 \neq 0$, $\beta_1 > 0$, $\beta_1 < 0$) слід використовувати односторонні критичні множини типу (u, $+\infty$). (Тільки при перевірці першої з цих гіпотез в якості граничної точки u слід брати квантиль розподілу $F_{1,n}$ рівня $1 - \alpha$, а при перевірці двох інших гіпотез — аналогічний квантиль рівня $1 - \alpha / 2$, де α — рівень значущості критерію). По-друге, ми бачимо, що обидва згадані способи перевірки гіпотези про незначимість регресії є, по суті, еквівалентними. Використання обох цих способів у статистичному комп'ютерному забезпеченні пояснюється тим, що при розгляданні лінійної регресії з *кількома* незалежними змінними T-статистики служать для перевірки значущості *окремих* коефіцієнтів регресії, а F-статистики — для перевірки значущості регресії *в цілому*.

2.2.4.3.7. Вправа. Довести, що для простої лінійної регресії відношення F суми квадратів, що обумовлена регресією (інакше, обумовлена моделлю – рівність (3.3)) до суми квадратів відносно регресії має розподіл Фішера

$F_{1,\,n-2}$.

Вказівка. Величина у лівій частині рівності (3.3), поділена на σ^2, має розподіл $\chi^2(n–1)$. За теоремою 4.2.1 перший доданок у правій її частині, поділений на σ^2, розподілений за законом $\chi^2(n–2)$. Тому за теоремою 4.3.3 другий доданок з правої частини (3.2), поділений на σ^2, має розподіл $\chi^2(1)$ і не залежить від першого доданку. Звідси розглядуване відношення

$$F = \frac{\sum(\hat{y}_i - \bar{y})^2}{S^2} \tag{4.27}$$

має розподіл $F_{1,\,n-2}$.

2.2.4.3.8. Зауваження. За означенням b_1,

$$F = \frac{b_1^2 \sum(x_i - \bar{x})^2}{S^2} = \left[\frac{b_1\left(\sum(x_i - \bar{x})^2\right)^{1/2}}{S}\right]^2.$$

Дана величина є квадратом статистики $T(b_1)$ при β_1=0 (рівність (4.19)). Звідси випливає, що F-відношення (4.27) дійсно може використовуватися як статистика критерію при перевірці гіпотези про незначущість коефіцієнту регресії

2.5. Загальна лінійна модель та оцінювання її параметрів за МНК

Припустимо, що залежна змінна y з точністю до випадкової адитивної похибки ε може бути представлена як лінійна комбінація введених вище факторних змінних (незалежних змінних, регресорів) x_0, x_1,…, x_{p-1}:

$$y = \beta_0 x_0 + \ldots + \beta_{p-1} x_{p-1} + \varepsilon, \tag{3.1}$$

де коефіцієнти $\beta_0,\ldots, \beta_{p-1}$ є невідомими. Для конкретності вважатимемо надалі, що змінні $x_0, x_1,\ldots, x_{p-1}$ та y набувають дійсних значень.

Припустимо, що маємо вибірку об'єму n, яка являє собою сукупність одержаних експериментальним шляхом n наборів чисел вигляду

$$(x_{i0},\ldots,x_{i,p-1}, y_i),\ i = 1,2,\ldots,n, \tag{3.2}$$

де x_{ij} — значення j-го регресора (j-ї незалежної змінної) при i-му спостереженні, y_i — відповідне значення залежної змінної y. Значення похибки ε при i-му спостереженні позначимо ε_i. Підставляючи послідовно вибіркові значення (3.2) в рівність (3.1), одержимо n рівностей:

$$y_i = \beta_0 x_{i0} + \ldots + \beta_{p-1} x_{i,p-1} + \varepsilon_i,\ i = 1,2,\ldots, n. \tag{3.3}$$

Введемо наступні векторно-матричні позначення:

$$\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} x_{10} & x_{11} & \ldots & x_{1,p-1} \\ x_{20} & x_{21} & \ldots & x_{2,p-1} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n0} & x_{n1} & \ldots & x_{n,p-1} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \ldots \\ \beta_{p-1} \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \ldots \\ \varepsilon_n \end{pmatrix},$$

Таким чином: $\boldsymbol{Y}$ — вектор спостережень залежної змінної, він має розмірність n (n-вектор); $\boldsymbol{X}$ — матриця спостережень незалежних змінних розміру ($n \times p$) (($n\times p$)- матриця), вона носить назву регресійної матриці або матриці експерименту, $\boldsymbol{\beta}$ — p-вектор (невідомих) коефіцієнтів, $\boldsymbol{\varepsilon}$ — n-вектор помилок. За допомогою цих позначень систему рівностей (3.3) можна записати у вигляді однієї векторно-матричної рівності

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}, \tag{3.4}$$

Зауважимо, що з метою кращого розрізняння скалярних і векторно-матричних величин для останніх тут використовується напівжирний шрифт. Крім того, надалі використовується знак $'$ як символ транспонування матриці або вектора, зокрема, вектор-стовпець $\boldsymbol{Y}$ може бути записаним як $(y_1,\ldots,y_n)'$.

Модель залежності даних у формах (3.1) — (3.4) має назву *лінійної регресійної моделі*, причому залежність у формі (3.4) часто зветься *загальною лінійною моделлю* (ЗЛМ) [1-7]. Функція

$$f(x) = \boldsymbol{\beta}'\boldsymbol{x} = \beta_0 x_0 + \ldots + \beta_{p-1} x_{p-1}$$

має назву (лінійної) *функції регресії*.

Для оцінювання доцільності використанні лінійної моделі при описі досліджуваної залежності можна використовувати так званий *множинний коефіцієнт кореляції*. Множинний коефіцієнт кореляції $\rho_{y\cdot x}$ між залежною змінною y і предикторними (незалежними) змінними x_0, x_1, …, x_{p-1} є максимумом значень парних (звичайних) коефіцієнтів кореляції між y і всілякими лінійними комбінаціями x_0, $x_1,\ldots,x_{p-1}$. Лінійна комбінація, на якій досягається зазначений максимум, є оптимальним лінійним предиктором змінної y за змінними x_0, $x_1,\ldots$ в тому сенсі, що на згаданій лінійній комбінації досягається мінімальне можливе значення величин $M[L(x)-y]^2$, де M — символ математичного сподівання, а $L(x)$ пробігає множину всіх лінійних комбінацій x_0, $x_1,\ldots,x_{p-1}$.

Якщо відома коваріаційна матриця $\boldsymbol{\Sigma}_{\boldsymbol{Z}}$ вектора $\boldsymbol{Z} = (y, x_0, x_1,\ldots,x_{p-1})$:

$$\boldsymbol{\Sigma}_{\boldsymbol{Z}} = \begin{pmatrix} \sigma_y^2 & \boldsymbol{\Sigma}_{yx} \\ \boldsymbol{\Sigma}_{xy} & \boldsymbol{\Sigma}_{x} \end{pmatrix}, \qquad (3.5)$$

де σ^2_y — дисперсія y, $\mathbf{\Sigma}_{yx}$ — вектор-рядок коваріацій між y та $x_0, x_1,\ldots,x_{p-1}$, $\mathbf{\Sigma}_{xy}$ —вектор-стовпець аналогічних коваріацій, $\mathbf{\Sigma}_x$ — коваріаційна матриця вектора $\boldsymbol{x} = (x_0, x_1,\ldots,x_{p-1})'$, то множинний коефіцієнт кореляції $\rho_{y\cdot x}$ може бути знайденим за формулою

$$\rho_{y\cdot x} = (\mathbf{\Sigma}_{yx}\,\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy})^{1/2} / \sigma_y. \tag{3.6}$$

На практиці замість величини $\rho_{y\cdot x}$ обчислюється її оцінка $r_{y\cdot x}$:

$$r_{y\cdot x} = (\boldsymbol{S}_{yx}\,\boldsymbol{S}_x^{-1}\boldsymbol{S}_{xy})^{1/2} / s_y\,, \tag{3.7}$$

де $\boldsymbol{S}_{yx}$, $\boldsymbol{S}_x$, $\boldsymbol{S}_{xy}$ та s_y — оцінки, відповідно, $\mathbf{\Sigma}_{yx}$ $\mathbf{\Sigma}_x$, $\mathbf{\Sigma}_{xy}$ та σ_y: якщо x_{ij} — i-те вибіркове значення змінної x_j, $1 \le i \le n$, $0 \le j \le p-1$, y_i — i-те вибіркове значення змінної y, n — кількість вибіркових значень, то (i, j)-й елемент матриці $\boldsymbol{S}_x$ і j-й елемент вектора-рядка $\boldsymbol{S}_{yx}$ дорівнюють, відповідно

дорівнює

$$\frac{1}{n-1}\sum_{k=1}^{n}(x_{ki}-\bar{x}_i)(x_{kj}-\bar{x}_j)\,, \quad \frac{1}{n-1}\sum_{k=1}^{n}(y_k-\bar{y})(x_{kj}-\bar{x}_j)$$

при

$$\bar{x}_j = \frac{1}{n}\sum_{k=1}^{n}x_{kj}\,,\ \bar{y} = \frac{1}{n}\sum_{k=1}^{n}y_k\,,\ i = 1,\ldots,n,\ j = 0,\ldots,p-1$$

Чим ближче величина $\rho_{y\cdot x}$ до 1, тим вищим є степінь лінійної залежності між y і змінними $x_0, x_1,\ldots,x_{p-1}$. Для перевірки гіпотези про відсутність лінійної залежності можна використати наступний вираз (F-відношення):

$$F = \frac{n-p-1}{p}\frac{r_{y\cdot x}^2}{1-r_{y\cdot x}^2}. \qquad (3.8)$$

Гіпотеза про відсутність лінійної залежності відхиляється на рівні значущості α, якщо величина F виявляється більшою, ніж квантиль рівня $1-\alpha$ розподілу Фішера $F(p, n-p-1)$ [1-7,13]. Зауважимо, що вказана процедура перевірки є повністю обґрунтованою, якщо права частина рівності (3.8) розподілена за законом Фішера $F(p, n-p-1)$. Такий розподіл напевне має місце, якщо розглядувана модель є не тільки лінійною, а ще й нормальною (лінійно-нормальна модель), тобто якщо величина похибки ε має нормальний розподіл з нульовим середнім значенням. У першому наближенні зазначений спосіб перевірки лінійності може застосовуватися і без гарантій наявності такого розподілу у величини ε, оскільки розподіл Фішера має певні властивості стійкості по відношенню до відхилень від нормальності даних [3,4]/

Зауважимо, що за наявністю явного вигляду лінійної моделі, побудованої за методом найменших квадратів, перевірка доцільності її використання, замість обчислень за рівностями (3.7), (3.8) може бути виконана за допомогою так званого вибіркового коефіцієнта множинної кореляції в рамках описаної нижче процедури перевірки гіпотези про незначущість лінійної регресійної моделі.

За прийняттям моделі (3.1) — (3.4) для опису досліджуваних даних першою задачею становиться оцінка вектора коефіцієнтів $\boldsymbol{\beta}$ з використанням результатів спостережень факторних змінних $x_0, x_1,\ldots, x_{p-1}$ і змінної відгуку y. Найпоширенішим методом такої оцінки в регресійних моделях є метод найменших квадратів (МНК). Згідно з цим методом в якості оцінки вектора

коефіцієнтів $\boldsymbol{\beta}$ приймається вектор $\boldsymbol{b} = (b_0,\ldots,b_{p-1})'$, що є розв'язком задачі мінімізації

$$\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \|^2 \xrightarrow[\beta \in R^p]{} min\ , \tag{3.9}$$

де R^p — p-вимірний евклідів простір, мінімум розшукується по $\boldsymbol{\beta} \in R^p$, а норма $\|z\|$ вектора $z = (z_1,\ldots,z_n) \in R^n$ визначена як $(z_1^2 + \ldots + z_n^2)^{1/2}$. Відомо [1-6], що розв'язок задачі (3.9) можна знайти як розв'язок $\boldsymbol{b}$ системи так званих нормальних рівнянь

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}'\boldsymbol{Y}. \tag{3.10}$$

За будь-яких умов система (3.10) є сумісною, причому у випадку, коли матриця $\boldsymbol{X}$ має повний ранг, її розв'язок (отже, і розв'язок задачі (3.9)) єдиний і має вигляд

$$\boldsymbol{b} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}. \tag{3.11}$$

Оцінка $\boldsymbol{b}$ вектора $\boldsymbol{\beta}$ за МНК буде скорочено називатися МНК-оцінкою.

Незалежно від того, є розв'язок системи (3.10) єдиним чи ні, величина

$$RSS := \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{b}\|^2 \tag{3.12}$$

приймає одне й те ж саме значення, який би розв'язок $\boldsymbol{b}$ зазначеної системи не був підставлений у праву частину рівності (3.12). (Зауважимо, що *RSS* — абревіатура від англійського residual sum of squares, тобто залишкова сума квадратів.)

Очевидно, має місце рівність

$$RSS = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{b})'\,(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{b}), \tag{3.13}$$

Інколи буває зручнішими інші вирази для *RSS*, наприклад,

$$RSS = \boldsymbol{Y}'\boldsymbol{Y} - \boldsymbol{b}'\boldsymbol{X}'\boldsymbol{Y} = \boldsymbol{Y}'\boldsymbol{Y} - \boldsymbol{b}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{b}, \tag{3.14}$$

які неважко отримати з (3.13), враховуючи рівність (3.10).

Деякі ймовірнісні припущення та їх наслідки

За прийняттям деяких припущень ймовірнісного характеру з'являється можливість дати відповідь на питання щодо якості обраної моделі і можливості її використання. У більшості випадків такі відповіді базуються на можливості побудови довірчих інтервалів для параметрів, що оцінюються, і можливості перевірки різноманітних статистичних гіпотез.

Наступні припущення є характерними для класичного регресійного аналізу [3,4].

1) похибки визначення незалежних змінних x_j відсутні або несуттєві;
2) середнє значення випадкової складової змінної y дорівнює 0, зокрема, для моделі (8.1) $M\varepsilon = M\varepsilon_i = 0$, $i = 1,\ldots, n$ (M — символ математичного сподівання випадкової величини або вектора);
3) значення ε_i є попарно некорельованими і мають однакову скінченну дисперсію: $D\varepsilon = D\varepsilon_i = \sigma^2 = const$, $i = 1,\ldots,n$ (D — символ дисперсії випадкової величини);
4) матриця $\boldsymbol{X}$ має ранг p (тобто є матрицею повного рангу);
5) вектор помилок $\boldsymbol{\varepsilon}$ (див. (8.2)) має n-вимірний нормальний розподіл ймовірностей $\boldsymbol{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n)$, тобто n-вимірний нормальний розподіл з вектором середніх $\boldsymbol{0} = (0,\ldots,0)'$ та коваріаційною матрицею $\sigma^2\boldsymbol{I}_n$, де $\boldsymbol{I}_n$ — одинична матриця (діагональна матриця, всі діагональні елементи якої дорівнюють 1). Еквівалентне формулювання даного

припущення: $\boldsymbol{Y}$ має n-вимірний нормальний розподіл з середнім значенням $\boldsymbol{X\beta}$ та коваріаційною матрицею $\sigma^2\boldsymbol{I}_n$ (коротше, $\boldsymbol{Y} \in \boldsymbol{N}(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I}_n)$).

Зазначені припущення гарантують наявність важливих імовірнісно-статистичних властивостей визначених вище оцінок вектора $\boldsymbol{\beta}$. Так, за виконанням вимог 1) — 3) лінійна модель (8.4), очевидно, може бути подана у вигляді

$$M\boldsymbol{Y} = \boldsymbol{X\beta}.$$

Якщо при цьому виконується ще й припущення 4), то неважко переконатися, що МНК-оцінка $\boldsymbol{b}$ вектора $\boldsymbol{\beta}$ є незміщеною [дод]:

$$M\boldsymbol{b} = \boldsymbol{\beta}.$$

За виконанням всіх вимог 1) — 5) МНК-оцінка $\boldsymbol{b} = (b_0,\ldots,b_{p-1})$ є ефективною в класі всіх лінійних за $\boldsymbol{Y}$ незміщених оцінок вектора $\boldsymbol{\beta}$, тобто для довільної такої оцінки $\tilde{b} = (\tilde{b}_0,\ldots, \tilde{b}_{p-1})$ мають місце нерівності $D(b_j) \le D(\tilde{b}_j)$, $j = 0,\ldots, p-1$, де D — символ дисперсії випадкової величини. При цьому кожна оцінка, що є ефективною в означеному класі оцінок, є МНК-оцінкою (теорема Гаусса — Маркова [4]). За певних додаткових умов (які тут не наводяться — див. [3,4]) МНК-оцінки $\boldsymbol{b} = \boldsymbol{b}_n$ є обґрунтованими оцінками параметру $\boldsymbol{\beta}$, тобто такими, що збігаються до $\boldsymbol{\beta}$ за ймовірністю:

$$\forall\ \delta > 0\ P(\|\boldsymbol{b}_n - \boldsymbol{\beta}\| > \delta) \to 0 \text{ при } n \to \infty,$$

де P є символом ймовірності, $\|\cdot\|$ — визначена вище норма в просторі R^p.

Припущення 5) не відіграє ніякої ролі при знаходженні МНК-оцінки $\boldsymbol{\beta}$, але за його прийняттям з'являються додаткові приводи для використання

МНК. По-перше, оцінка $\boldsymbol{\beta}$ за МНК співпадає у даному випадку з його оцінкою максимальної правдоподібності (ММП). По-друге, з'являються можливості визначення розподілів певних статистик, що залежать від значень регресорів x_j та залежної змінної y. Останнє дає змогу ефективного знаходження довірчих інтервалів для тих чи інших параметрів регресії та виконання перевірки різноманітних статистичних гіпотез, що пов'язані з якістю побудованої моделі. Вказані можливості багато в чому засновані на наступній теоремі, яку треба розуміти так, що коли є істинною ЗЛМ (8.2) (з прийняттям припущень 1) — 5)), то мають місце твердження зазначеної теореми. При її формулюванні використовуються позначення: χ^2_n — розподіл «хі-квадрат» з n степенями свободи; $\boldsymbol{N_p}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ — p-вимірний нормальний розподіл з вектором середніх значень $\boldsymbol{\mu}$ і коваріаційною матрицею $\boldsymbol{\Sigma}$ [3,4];

Теорема 1 (про ймовірнісні властивості оцінок МНК).

1. оцінка $\boldsymbol{b}$ вектора параметрів $\boldsymbol{\beta}$ має розподіл $\boldsymbol{N_p}(\boldsymbol{\beta},\sigma^2(\boldsymbol{X'X})^{-1})$;
2. квадратична форма $(\boldsymbol{b}-\boldsymbol{\beta})'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{b}-\boldsymbol{\beta})/\sigma^2$ має розподіл χ^2_p;
3. вектор оцінок $\boldsymbol{b}$ і величина S^2 незалежні одне від одного;
4. величина RSS/σ^2 має розподіл χ^2_{n-p};
5. величина $S^2 := RSS/(n-p)$ є незміщеною оцінкою дисперсії похибки спостережень σ^2;

Теорема 1 має важливі наслідки в лінійному регресійному аналізі, але її застосування є строго обґрунтованим лише у випадку адекватності лінійної моделі (3.4) тій залежності, що представлена вказаними вище вибірковими даними (3.2).

Деякі ймовірнісні припущення та їх наслідки

За прийняттям деяких припущень ймовірнісного характеру з'являється можливість дати відповідь на питання щодо якості обраної моделі і можливості її використання. У більшості випадків такі відповіді базуються на можливості побудови довірчих інтервалів для параметрів, що оцінюються, і можливості перевірки різноманітних статистичних гіпотез.

Наступні припущення є характерними для класичного регресійного аналізу [1-7]

6) похибки визначення незалежних змінних x_j відсутні або несуттєві;
7) середнє значення випадкової складової змінної y дорівнює 0, зокрема, для моделі (8.1) $M\varepsilon = M\varepsilon_i = 0$, $i = 1,\ldots, n$ (M — символ математичного сподівання випадкової величини або вектора);
8) значення ε_i є попарно некорельованими і мають однакову скінченну дисперсію: $D\varepsilon = D\varepsilon_i = \sigma^2 = const$, $i = 1,\ldots,n$ (D — символ дисперсії випадкової величини);
9) матриця $\boldsymbol{X}$ має ранг p (тобто є матрицею повного рангу);
10) вектор помилок $\boldsymbol{\varepsilon}$ (див. (8.2)) має n-вимірний нормальний розподіл ймовірностей $\boldsymbol{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n)$, тобто n-вимірний нормальний розподіл з вектором середніх $\boldsymbol{0} = (0,\ldots,0)'$ та коваріаційною матрицею $\sigma^2\boldsymbol{I}_n$, де $\boldsymbol{I}_n$ — одинична матриця (діагональна матриця, всі діагональні елементи якої дорівнюють 1). Еквівалентне формулювання даного припущення: $\boldsymbol{Y}$ має n-вимірний нормальний розподіл з середнім значенням $\boldsymbol{X\beta}$ та коваріаційною матрицею $\sigma^2\boldsymbol{I}_n$ (коротше, $\boldsymbol{Y} \in \boldsymbol{N}(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I}_n)$).

Зазначені припущення гарантують наявність важливих імовірнісно-статистичних властивостей визначених вище оцінок вектора $\boldsymbol{\beta}$. Так, за

виконанням вимог 1) — 3) лінійна модель (8.4), очевидно, може бути подана у вигляді

$$M\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta}.$$

Якщо при цьому виконується ще й припущення 4), то неважко переконатися, що МНК-оцінка $\boldsymbol{b}$ вектора $\boldsymbol{\beta}$ є незміщеною [дод]:

$$M\boldsymbol{b} = \boldsymbol{\beta}.$$

За виконанням всіх вимог 1) — 5) МНК-оцінка $\boldsymbol{b} = (b_0,\ldots,b_{p-1})$ є ефективною в класі всіх лінійних за $\boldsymbol{Y}$ незміщених оцінок вектора $\boldsymbol{\beta}$, тобто для довільної такої оцінки $\tilde{b} = (\tilde{b}_0,\ldots, \tilde{b}_{p-1})$ мають місце нерівності $D(b_j) \le D(\tilde{b}_j)$, $j = 0,\ldots, p-1$, де D — символ дисперсії випадкової величини. При цьому кожна оцінка, що є ефективною в означеному класі оцінок, є МНК-оцінкою (теорема Гаусса — Маркова [36]). За певних додаткових умов (які тут не наводяться — див. [4]) МНК-оцінки $\boldsymbol{b} = \boldsymbol{b}_n$ є обґрунтованими оцінками параметру $\boldsymbol{\beta}$, тобто такими, що збігаються до $\boldsymbol{\beta}$ за ймовірністю:

$$\forall\ \delta > 0\ P(\|\boldsymbol{b}_n - \boldsymbol{\beta}\| > \delta) \to 0 \text{ при } n \to \infty,$$

де P є символом ймовірності, $\|\cdot\|$ — визначена вище норма в просторі R^p.

Припущення 5) не відіграє ніякої ролі при знаходженні МНК-оцінки $\boldsymbol{\beta}$, але за його прийняттям з'являються додаткові приводи для використання МНК. По-перше, оцінка $\boldsymbol{\beta}$ за МНК співпадає у даному випадку з його оцінкою максимальної правдоподібності (ММП). По-друге, з'являються можливості визначення розподілів певних статистик, що залежать від значень регресорів x_j та залежної змінної y. Останнє дає змогу ефективного знаходження довірчих інтервалів для тих чи інших параметрів регресії та виконання

перевірки різноманітних статистичних гіпотез, що пов'язані з якістю побудованої моделі. Вказані можливості багато в чому засновані на наступній теоремі, яку треба розуміти так, що коли є істинною ЗЛМ (8.2) (з прийняттям припущень 1) — 5)), то мають місце твердження зазначеної теореми. При її формулюванні використовуються позначення: χ^2_n — розподіл «хі-квадрат» з n степенями свободи; $\boldsymbol{N_p}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ — p-вимірний нормальний розподіл з вектором середніх значень $\boldsymbol{\mu}$ і коваріаційною матрицею $\boldsymbol{\Sigma}$ [3,4].

Теорема 1 (про ймовірнісні властивості оцінок МНК).

6. оцінка $\boldsymbol{b}$ вектора параметрів $\boldsymbol{\beta}$ має розподіл $\boldsymbol{N_p}(\boldsymbol{\beta},\sigma^2(\boldsymbol{X'X})^{-1})$;
7. квадратична форма $(\boldsymbol{b}-\boldsymbol{\beta})'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{b}-\boldsymbol{\beta})/\sigma^2$ має розподіл χ^2_p;
8. вектор оцінок $\boldsymbol{b}$ і величина S^2 незалежні одне від одного;
9. величина RSS/σ^2 має розподіл χ^2_{n-p};
10. величина $S^2 := RSS/(n-p)$ є незміщеною оцінкою дисперсії похибки спостережень σ^2;

Теорема 1 має важливі наслідки в лінійному регресійному аналізі, але її застосування є строго обґрунтованим лише у випадку адекватності лінійної моделі (3.4) тому процесу, що представлений вказаними вище вибірковими даними (3.2). Опишемо метод такої перевірки, котрий є узагальненням на випадок кількох змінних розглянутого у попередньому підрозділі методу для простої лінійної регресії.

Серед методів перевірки адекватності лінійної моделі даних досить поширеним є метод, що полягає в порівнянні оцінок дисперсій похибок, що одержані, з одного боку, з застосуванням даної моделі, а з іншого — незалежним шляхом. Реалізація зазначеного методу полягає в наступному. Позначимо для даних (3.2) буквою $\boldsymbol{x}_i$ i-ту точку спостережень (вектор-рядок)

незалежної змінної, тобто $\boldsymbol{x}_i = (x_{i1},\ldots,x_{i,\,p-1})$, $i = 1,2,\ldots,n$. Обговорюваний метод потребує наявності кількох спостережень за y принаймні в одній з точок $\boldsymbol{x}_i$. Припустимо, що ця вимога виконується. Кажучи іншими словами, серед точок $\boldsymbol{x}_i$ є такі, що повторюються. Тоді нехай $\boldsymbol{x}^1, \boldsymbol{x}^2,\ldots, \boldsymbol{x}^m$ — *різні* точки спостережень (вектори-рядки), причому хоч в одній з них кількість спостережень більше ніж 1. Припустимо, що при $\boldsymbol{x} = \boldsymbol{x}^i$ спостерігалися значення $y_{i1}, \ldots. y_{i n_i}$, $i = 1,\ldots., m$ (ясно, що $n = n_1 + \ldots + n_m$). Нехай $\hat{y}$ — *оцінка функції регресії*, тобто

$$\hat{y} = b_0 x_0 + \ldots + b_{p-1} x_{p-1}. \tag{3.15}$$

Позначимо

$$\hat{y}_i = \hat{y}(\boldsymbol{x}^i),\ \bar{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}\ \ (i = 1,\ldots, m),$$

$$S_1^{\,2} = \frac{1}{m-p}\sum_{i=1}^{m} n_i(\hat{y}_i - \bar{y}_i)^2,\ S_2^{\,2} = \frac{1}{n-m}\sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_i)^2.$$

Тоді, якщо дисперсії y при кожному $\boldsymbol{x}^i$ можна вважати рівними між собою та $m > p$, то відношення вигляду $S_1^{\,2} / S_2^{\,2}$ (варіант з сукупності так званих F-відношень) має розподіл Фішера $F(m - p, n - m)$, причому гіпотеза про адекватність моделі $\hat{y}$ *не приймається* на рівні α, якщо вказане відношення перевищує квантиль рівня $1 - \alpha$ вказаного розподілу. У іншому випадку гіпотеза приймається [3,4].

Зауваження 1. За наявністю точок повторних спостережень, про які щойно йшла мова, системі нормальних рівнянь (3.10) і її розв'язку (3.11) можна надати спеціального вигляду. Так, позначимо $\overline{\boldsymbol{X}}$ матрицю, перший,…, m-й

рядки якої утворені векторами $\boldsymbol{x}^1$, $\boldsymbol{x}^2$,…, $\boldsymbol{x}^m$ відповідно, $\boldsymbol{N}$ — діагональну матрицю порядку $m \times m$ вигляду

$$\boldsymbol{N} = \begin{pmatrix} n_1 & 0 & \dots & 0 \\ 0 & n_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & n_m \end{pmatrix},$$

в якій n_j — кількість повторних спостережень в точці $\boldsymbol{x}^j$, $\overline{\boldsymbol{Y}}$ — вектор, кожна j- та координата якого дорівнює середньому арифметичному спостережень залежної змінної в точці $\boldsymbol{x}^j$, $j = 1,\dots, m$. Тоді система рівнянь (3.10) запишеться у вигляді

$$\overline{\boldsymbol{X}}'\boldsymbol{N}\overline{\boldsymbol{X}}\,\boldsymbol{\beta} = \overline{\boldsymbol{X}}'\boldsymbol{N}\,\overline{\boldsymbol{Y}}$$

і за умови повноти рангу матриці $\overline{\boldsymbol{X}}$ її розв'язок матиме вигляд

$$\boldsymbol{b} = (\overline{\boldsymbol{X}}'\boldsymbol{N}\overline{\boldsymbol{X}})^{-1}\overline{\boldsymbol{X}}'\boldsymbol{N}\overline{\boldsymbol{Y}}. \qquad (3.16)$$

Зокрема, при $n_1 = \dots = n_m$ одержимо

$$\boldsymbol{b} = (\overline{\boldsymbol{X}}'\overline{\boldsymbol{X}})^{-1}\overline{\boldsymbol{X}}'\overline{\boldsymbol{Y}}. \qquad (3.16)'$$

Надалі вважається, що виконуються сформульовані вище припущення 1) — 5) (реально це може означати, що лінійна модель (3.4) пройшла перевірку на адекватність). Щодо вибіркових даних (3.2) то тут буде припускатися, що вибірка має достатньо великий об'єм, а саме, що $n > p$.

Довірчі інтервали для параметрів регресії

Твердження даного підрозділу стосуються вигляду довірчих інтервалів для параметрів регресії і можуть бути одержані як наслідки теореми 1 [3,4].

Спочатку наведемо відомості щодо довірчих інтервалів для лінійних комбінацій невідомих коефіцієнтів регресії, зокрема, для самих вказаних коефіцієнтів. Нехай $\boldsymbol{a}$ — p-вимірний дійсний вектор. Позначимо $u^t(\alpha, n-p)$ квантиль рівня $1-\alpha/2$ розподілу Стьюдента з $n-p$ степенями свободи, $\hat{D}(\boldsymbol{a}'\boldsymbol{b}) = S^2[\boldsymbol{a}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{a}]$ (це — незміщена оцінка дисперсії величини $\boldsymbol{a}'\boldsymbol{b}$; S^2 визначено у п.5 теореми 1). Тоді інтервал $J(\boldsymbol{a},\alpha)$ з кінцями

$$\boldsymbol{a}'\boldsymbol{b} \pm u^t(\alpha, n-p)\sqrt{\hat{D}(\boldsymbol{a}'\boldsymbol{b})}, \tag{3.17}$$

є довірчим інтервалом рівня α для параметру $\boldsymbol{a}'\boldsymbol{\beta}$. Зокрема, довірчий інтервал рівня α для параметру β_j, $0 \le j \le p-1$, одержимо, поклавши $\boldsymbol{a} = (0,\ldots,0,1,0,\ldots,0)$, де 1 стоїть на $(j+1)$- му місці.

Зауважимо, що прямий добуток по індексу l інтервалів $J(\boldsymbol{a}_l, \alpha)$, $1 \le l \le k$, не утворює довірчу множину рівня α для векторного параметра $(\boldsymbol{a}_1'\boldsymbol{b},\ldots,\boldsymbol{a}_k'\boldsymbol{b})$ [35]. Рівень значущості, не більший за α, забезпечить зазначений прямий добуток інтервалів $J(\boldsymbol{a}_l, \alpha/k)$.

Якщо вектор коефіцієнтів $\boldsymbol{\beta}$ є оціненим за вибірковими даними (3.2) то оцінку функцію регресії $\hat{y} = \hat{y}(\boldsymbol{x})$ (рівність (3.15)) можна використати для дослідження форми поверхні регресії

$$f(x_0,\ldots,x_{p-1}) = \beta_0 x_0 + \ldots + \beta_{p-1} x_{p-1}$$

Зокрема, можна побудувати довірчі інтервали для значень функції f при різних значеннях аргументу $\boldsymbol{x} = (x_0,\ldots,x_{p-1})$. Такі інтервали мають вигляд

$$\hat{y}(\boldsymbol{x}) \pm u^t(\alpha, n-p)\, S(v)^{1/2}, \tag{3.18}$$

де $v = \boldsymbol{x}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}$, $S = (S^2)^{1/2}$, тобто в точності вигляд (3.17) при $\boldsymbol{a} = \boldsymbol{x}$ (оскільки $f(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\beta}$, $\hat{y}(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{b}$).

На практиці довірчі інтервали для значень функції відгуку

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \varepsilon$$

часто становлять навіть більший інтерес, ніж довірчі інтервали для значень функції регресії $f(\boldsymbol{x})$. При фіксованому $\boldsymbol{x}$ довірчий інтервал для $y(\boldsymbol{x})$ має вигляд [33, 35]:

$$\hat{y}(\boldsymbol{x}) \pm u^t(\alpha, n-p)\, S\,(v+1)^{1/2}, \tag{3.19}$$

де зміст позначень такий самий, що й у рівності (3.18).

МНК за наявністю лінійних обмежень і перевірка гіпотез для лінійної регресії

Нехай для опису даних прийнято модель (3.4) з додатковою умовою

$$\boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{c}, \tag{3.20}$$

де $\boldsymbol{A}$ і $\boldsymbol{c}$ — відомі матриця і вектор відповідно. Надалі для скорочення умова (3.20) інколи фігурує як «умова H», а в наступному підрозділі — як «гіпотеза H». Принцип найменших квадратів полягає в даному випадку в визначенні оцінки $\boldsymbol{b}_H$ вектору параметрів $\boldsymbol{\beta}$ як розв'язку задачі мінімізації величини $\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \|^2$ з урахуванням умови H (3.20)

$$\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \|^2 \xrightarrow[\boldsymbol{\beta}:\boldsymbol{A}\boldsymbol{\beta}=\boldsymbol{c}]{} min\,, \tag{3.21}$$

Для розв'язання даної задачі можна застосувати відомий метод невизначених множників Лагранжа, згідно з яким мінімізується вираз

$$(\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{\beta}) + \boldsymbol{\lambda}'(\boldsymbol{c} - \boldsymbol{A}\,\boldsymbol{\beta}) \tag{3.22}$$

за (векторними) параметрами $\boldsymbol{\lambda}$ та $\boldsymbol{\beta}$. Розв'язок відносно $\boldsymbol{\beta}$ цієї задачі і становить шуканий вектор параметрів $\boldsymbol{b}_H$. Якщо матриці $\boldsymbol{X}$ і $\boldsymbol{A}$ мають повний ранг, то

$$\boldsymbol{b}_H = \boldsymbol{b} + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{A}'[\boldsymbol{A}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{A}']^{-1}(\boldsymbol{c} - \boldsymbol{A}\,\boldsymbol{b}) \tag{3.23}$$

(див. [3,4]).

Надалі, поряд з абревіатурою *R S S* (рівність (3.12)) використовується позначення RSS_H — залишкова сума квадратів при виконанні обмеження (3.20):

$$RSS_H = \| \boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{b}_H \|^2 = (\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{b}_H)'\,(\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{b}_H). \tag{3.24}$$

Розглянемо лінійну модель

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3.25}$$

в якій $n > p$, $\boldsymbol{X} = \boldsymbol{X}_{n \times p}$ є матрицею повного рангу, тобто $r(\boldsymbol{X}) = p$ (нижні індекси матриці $\boldsymbol{X}$ вказують її розміри, а символ $r(\boldsymbol{X})$ означає ранг матриці). Припустимо, що треба перевірити гіпотезу

$$H\colon \boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{c}, \tag{3.26}$$

де $\boldsymbol{A}$ — відома матриця розміру $q \times p$ рангу q, а $\boldsymbol{c}$ — відомий q-вектор. В регресійному аналізі гіпотеза H носить назву *загальної лінійної гіпотези* (ЗЛГ). Основні дії по її перевірці цієї гіпотези базуються на наступній теоремі [3,4].

Теорема 2. Статистика

$$F = \frac{(RSS_H - RSS)/q}{RSS/(n-p)} \tag{3.27}$$

а) дорівнює

$$(\boldsymbol{Ab} - \boldsymbol{c})'[\boldsymbol{A}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{A}']^{-1}(\boldsymbol{A}\,\boldsymbol{b} - \boldsymbol{c})/(q\,S^2); \tag{3.28}$$

б) за умови вірності гіпотези H має розподіл $F_{q,\,n-p}$ (F- розподіл Фішера з (q, $n-p$) степенями свободи (див. додаток 15). З теореми 2 випливає

Процедура перевірки лінійної гіпотези H для загальної лінійної моделі регресії:

1) обчислюється F- відношення (3.27);

2) вказане F- відношення порівнюється з квантилем $u^F(\alpha; q, n-p)$ рівня $1-\alpha$ розподілу Фішера $F_{q,\,n-p}$; гіпотеза H відхиляється на рівні значущості α, якщо $F \geq u^F(\alpha; q, n-p)$, в протилежному випадку гіпотеза H відхиляється.

Зауваження 3. Таким чином, даний F- критерій перевірки ЗЛГ є одностороннім. Це обґрунтовується тим, що, з одного боку за умови вірності гіпотези H величини RSS і RSS_H повинні бути близькими одна до одної, а з другого тим, що величина (3.27) є невід'ємною (очевидно, $RSS_H \geq RSS$).

Зауважимо також, що для обчислення F- відношення не обов'язково користуватися рівністю (3.28): досить часто дане відношення припускає безпосереднє обчислення, яке полягає в явному знаходженні належних мінімальних значень в задачах (3.9) і (3.21).

Багато конкретних процедур перевірки гіпотез, що часто застосовуються на практиці, є частковими випадками щойно викладеної схеми. Розглянемо декілька прикладів з даного приводу.

Приклад 1. *Порівняння середніх значень двох нормальних сукупностей*.

Нехай $u_1, u_2,\ldots, u_{n_1}, v_1, v_2,\ldots, v_{n_2}$ — дві вибірки (як звичайно, вибіркові значення вважаються незалежними випадковими величинами). За умовою, величини u_j мають теоретичний розподіл $N(\mu_1, \sigma^2)$, а величини v_j — розподіл $N(\mu_2, \sigma^2)$. Гіпотеза H полягає в тому, що $\mu_1 = \mu_2$.

Ситуації, в яких виникає необхідність перевірки такої гіпотези, часто зустрічається при аналізі результатів механічних випробувань, течії виробничих процесів, процесів економіки і т.п.

Помітимо, що гіпотеза H може бути сформульована як деяка лінійна гіпотеза для лінійної регресійної моделі. Дійсно, за умовою

$$u_j = \mu_1 + \varepsilon_j, j = 1,2,\ldots, n_1, \tag{3.29}$$

$$v_j = \mu_2 + \varepsilon_{n_1 + j}, j = 1,2,\ldots, n_2. \tag{3.30}$$

Дані співвідношення можна переписати у векторно-матричному вигляді, якщо позначити

$$\boldsymbol{Y} = (u_1, u_2,\ldots, u_{n_1}, v_1, v_2,\ldots, v_{n_2})', \boldsymbol{\beta} = (\mu_1, \mu_2)',$$

$$\varepsilon = (\varepsilon_1, \varepsilon_2,\ldots, \varepsilon_{n_1}, \varepsilon_{n_1+1}, \varepsilon_{n_1+2},\ldots, \varepsilon_{n_1+n_2})',$$

$$\boldsymbol{X}' = \begin{pmatrix} 1 & 1 & \ldots & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \end{pmatrix}$$

(в першому рядку матриці $\boldsymbol{X}'$ міститься n_1 одиниць і n_2 нулі, а у другому — навпаки, n_1 нуль і n_2 одиниці). Тоді рівності (3.29), (3.30) запишуться у

вигляді (3.25). Гіпотеза H може бути записаною у вигляді (3.26), якщо покласти $\boldsymbol{A} = (1, -1), \boldsymbol{c} = \boldsymbol{0} = (0,0)'$.

Маємо $q = r(\boldsymbol{A}) = 1$,

$$\boldsymbol{X}'\boldsymbol{X} = \begin{pmatrix} n_1 & 0 \\ 0 & n_2 \end{pmatrix},$$

$$\boldsymbol{X}'\boldsymbol{X}^{-1} = \begin{pmatrix} \frac{1}{n_1} & 0 \\ 0 & \frac{1}{n_2} \end{pmatrix}.$$

Звідси неважко одержати, що

$$\boldsymbol{b} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y} = (\bar{u}, \bar{v})',$$

$$RSS = \boldsymbol{Y}'\boldsymbol{Y} - \boldsymbol{b}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{b} = \sum_i (u_i - \bar{u})^2 + \sum_j (v_j - \bar{v})^2.$$

де $\bar{u}, \bar{v}$ — вибіркові середні значення, відповідно, для змінних u_j та v_j, а для обчислення RSS використовувалася рівність (3.14). Підставляючи одержані вирази в (3.28), одержуємо, що

$$F = \frac{(\bar{u} - \bar{v})^2}{S^2(1/n_1 + 1/n_2)},$$

де $S^2 = RSS/(n - p) = RSS/(n_1 + n_2 - 2)$ (врахувати: $n = n_1 + n_2$, $p = 2$). Якщо гіпотеза H є вірною, то F має розподіл $F_{1,\ n_1+n_2-2}$. Правило приймання або відхилення гіпотез з таким розподілом F- статистик сформульоване вище.

Зауваження 2. Будь-яка випадкова величина, що має розподіл $F_{1,\ n}$, є квадратом випадкової величини, що має розподіл Стьюдента t_n [3,4].

Зокрема, звідси випливає, що гіпотезу прикладу 1 можна перевірити і за допомогою *t*- статистики:

$$t = \frac{\overline{u} - \overline{v}}{\sqrt{S^2(1/n_1 + 1/n_2)}} .$$

Гіпотеза H відхиляється на рівні значущості α, коли $|t| \geq u^t(\alpha, n_1 + n_2 - 2)$ (див. позначення до рівності (3.17)), у протилежному випадку не відхиляється.

Приклад 2. *Гіпотеза про незначущість лінійної регресії*.

Нехай математична модель залежності між y і $x_0, x_1, \ldots, x_{p-1}$ має вигляд

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_{p-1} x_{p-1} + \varepsilon, \tag{3.31}$$

(так що, зокрема, $x_0 \equiv 1$). Гіпотеза H має вигляд

$$\beta_1 = \ldots = \beta_{p-1} = 0.$$

Прийняття гіпотези H означає, що значення y, фактично, не залежить від тих значень, що приймають незалежні змінні, а такий стан речей свідчить про малу цінність даної моделі.

Для зведення ситуації до схеми ЗЛГ врахуємо, що запис моделі (3.31) після підстановки вибіркових даних (3.2), приймає вигляд (3.25) з матрицею експерименту

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1,p-1} \\ 1 & x_{21} & \ldots & x_{2,p-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_{n1} & \ldots & x_{n,p-1} \end{pmatrix}.$$

Введемо матрицю $\boldsymbol{A} = \boldsymbol{A}_{(p-1)\times p}$:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & . & . & 0 \\ 0 & 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 0 & 1 & 0 & . & 0 \\ . & . & . & . & . & . & . \\ 0 & . & . & . & 0 & 1 & 0 \\ 0 & . & . & . & . & 0 & 1 \end{pmatrix}.$$

Тоді наша гіпотеза H може бути записаною у вигляді (3.26) з нульовим (p – 1)- вимірним вектором $\boldsymbol{c}$. Згідно з рівністю (3.14) $RSS = \boldsymbol{Y}'\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{X}'\boldsymbol{Y}$, а

$$RSS_H = \min_{\beta_0} \sum_i (y_i - \beta_0)^2 = \sum_i (y_i - \bar{y})^2 = \boldsymbol{Y}'\boldsymbol{Y} - n\,\bar{y}^2,$$

де $\bar{y}$ — середнє арифметичне значень y_i, $i = 1,\ldots, n$. Тому, враховуючи, що $q = r(\boldsymbol{A}) = p - 1$, одержуємо

$$F = \frac{(RSS_H - RSS)/(p-1)}{RSS/(n-p)} = \frac{(\boldsymbol{b}'\boldsymbol{X}'\boldsymbol{Y} - n\bar{y}^2)}{(\boldsymbol{Y}'\boldsymbol{Y} - \boldsymbol{b}'\boldsymbol{X}'\boldsymbol{Y})} \cdot \frac{n-p}{p-1},$$

і статистика F має розподіл $F_{p-1,\,n-p}$, якщо гіпотеза H є вірною. Зауважимо, що за побудованою моделлю для перевірки даної гіпотези можна використовувати *вибірковий множинний коефіцієнт кореляції* R про який мова вже йшла вище. А саме, величина R визначається рівністю [3,4].

$$R = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{y})}{\left(\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2\right)^{1/2}},$$

а F-статистика для перевірки гіпотези H виражається через R наступним чином:$F = \frac{n-p}{p-1} \cdot \frac{R^2}{1-R^2}$. При справедливості H дана статистика має розподіл Фішера $F_{p-1,n-p}$ [3,4].

Приклад 3. *Гіпотеза про значення коефіцієнтів регресії*

Нехай має місце залежність (3.31). Розглянемо гіпотезу

$$H\colon \beta_j = c, \tag{3.32}$$

де c — задане число, $0 \le j \le p-1$. На практиці найчастіше потребує перевірки випадок $c = 0$, тобто гіпотеза H має вигляд

$$H\colon \beta_j = 0. \tag{3.33}$$

У такому випадку гіпотеза H має назву гіпотези про *незначущість коефіцієнту* β_j.

Гіпотеза H має вигляд (3.26), де $\boldsymbol{A}$ — матриця з одного рядка, всі елементи якої дорівнюють 0, крім елементу з номером $j+1$, який дорівнює 1. Звідси, якщо через $d_{j\,j}$ позначити $(j+1)$-й діагональний елемент матриці $(\boldsymbol{X}'\boldsymbol{X})^{-1}$, то $\boldsymbol{A}\,\boldsymbol{b} = b_j$, $\boldsymbol{A}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{A}' = d_{j\,j}$, і F- статистика для перевірки гіпотези H має вигляд

$$F = \frac{(b_j - c)^2}{S^2 d_{jj}}. \tag{3.34}$$

При справедливості гіпотези H статистика (3.34) має розподіл $F_{1,\,n-p}$. Як і в прикладі 2, F- статистика може розглядатися як квадрат відповідної t-статистики (зауваження 2).

Перевірка адекватності лінійної моделі регресії

Ставиться задача перевірки адекватності одне одному моделі (3.31) і наявних статистичних даних (дана задача з іншої точки зору розглядалася вище окремо для простої регресії і для загальної). Здається, неможливо надати тут терміну «адекватність» універсального змісту. У даному разі поняття «адекватність» означатиме добру узгодженість з деякою лінійною гіпотезою.

Припустимо, що серед наборів значень регресорів моделі (3.31), в яких виконуються спостереження змінної y, є хоч один, кількість спостережень в якому більше одиниці. Іншими словами можна сказати, що серед даних наборів незалежних змінних є такі, що повторюються. Тоді нехай $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m$ — *різні* набори спостережень (інакше, точки або вектори-рядки спостережень): $\boldsymbol{x}^i = (x_{i1}, \ldots, x_{i,r-1})$. Ми будемо вважати, що $m > p$. Позначимо n_i кількість спостережень в точці $\boldsymbol{x}^i$, n — загальну кількість спостережень, так що $n_1 + \ldots + n_m = n$ (хоч одне n_i більше 1). Виконуючи відповідні підстановки результатів спостережень в вираз (3.31), одержимо n рівнянь

$$y_{ir} = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_{p-1} x_{i,p-1} + \varepsilon_{ir}, \tag{3.35}$$

де y_{ir} — значення залежної змінної, $M\varepsilon_{ir} = 0$, $D\varepsilon_{ir} = \sigma^2$, $r = 1, \ldots, n_i$, $i = 1, \ldots. m$.

Позначимо

$$y_{ir} - \varepsilon_{ir} = z_i.$$

Зрозуміло, що коли модель (3.31) є істинною, то

$$z_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_{p-1} x_{i,p-1}, \; i = 1, \ldots, m. \tag{3.36}$$

Але якщо ми перевіряємо адекватність нашої моделі, то справедливість рівностей (3.36) є тільки гіпотезою. Надалі позначатимемо цю гіпотезу H. Зведемо її до деякої лінійної гіпотези, після чого скористуємося наведеною вище процедурою перевірки ЗЛГ.

Введемо наступні вектори і матриці: $\overline{\boldsymbol{X}}$ — матриця, перший,…, m-й рядки якої утворені векторами $(1, \boldsymbol{x}^1)$, $(1, \boldsymbol{x}^2)$,…, $(1, \boldsymbol{x}^m)$ відповідно $((1, \boldsymbol{x}^i) = (1, x_{i1},\ldots,x_{i,r-1}))$, $\boldsymbol{\beta} = (\beta_0, \beta_2,\ldots, \beta_{p-1})'$, $\boldsymbol{z} = (z_1,\ldots, z_m)'$,

$$\boldsymbol{\varepsilon} = (\varepsilon_{11}, \varepsilon_{12},\ldots, \varepsilon_{1,n_1},\ldots, \varepsilon_{m1},\ldots, \varepsilon_{m,\,n_m})',$$

$$\boldsymbol{Y} = (y_{11}, y_{12},\ldots, y_{1,\,n_1},\ldots, y_{m1},\ldots, y_{m,\,n_m})',$$

$$\boldsymbol{W} = \begin{pmatrix} \mathbf{1}_{n_1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{n_2} & \ldots & \mathbf{0} \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{1}_{n_m} \end{pmatrix}.$$

Пояснимо, що $\boldsymbol{W}$ — матриця, в якій $\mathbf{1}_{n_j}$ — вектори-стовпці розмірів n_j, що складені з одиниць (j = 1,.., m), $\mathbf{0}$ — вектори-стовпці відповідних розмірів, складені з нулів. Матриця $\boldsymbol{W}$ має розміри $n \times m$ і ранг m.

В обраних позначеннях зв'язок між вектором спостережень $\boldsymbol{Y}$ і вектором регресорів $\boldsymbol{z}$ має вигляд

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\varepsilon}, \tag{3.37}$$

а гіпотеза H може бути сформульована як

$$H\colon \boldsymbol{z} = \overline{\boldsymbol{X}}\,\boldsymbol{\beta}. \tag{3.38}$$

У даному вигляді гіпотеза H не має ще стандартного вигляду ЗЛМ, оскільки права частина рівності (3.38) є невідомою. Але все ж таки рівності (3.38) можна надати потрібної форми. Помітимо: рівність (8.38) означає, що вектор $\boldsymbol{z}$ належить векторному простору $R[\overline{\boldsymbol{X}}]$, що народжений стовпцями матриці $\overline{\boldsymbol{X}}$. Наступна лема дає зручну для нас умову такої належності.

Лема 1. Включення $\boldsymbol{z} \in R[\overline{\boldsymbol{X}}]$ має місце тоді і тільки тоді, коли для деякої матриці $A = A_{(m-p)\times m}$, $r(A) = m - p$, виконується рівність

$$\boldsymbol{Az} = \boldsymbol{0}. \tag{3.39}$$

Відзначимо, що в якості $\boldsymbol{A}$ можна взяти матрицю, складену з будь-яких $m - p$ лінійно незалежних рядків матриці $(\boldsymbol{I} - \boldsymbol{P})$, де $\boldsymbol{I} = \boldsymbol{I}_m$ — одинична $m \times m$ матриця, $\boldsymbol{P} = \overline{\boldsymbol{X}}(\overline{\boldsymbol{X}}'\overline{\boldsymbol{X}})^{-1}\overline{\boldsymbol{X}}'$ [35].

Таким чином, нашу задачу зведено до перевірки лінійної гіпотези (3.39) для лінійної моделі (3.37). В позначеннях вищенаведеної процедури перевірки треба покласти $q = m - p$, $n = n$, $p = m$, тому

$$F = \frac{(RSS_H - RSS)/(m-p)}{RSS/(n-m)}. \tag{3.40}$$

Безпосереднє обчислення RSS і RSS_H тут є більш зручним, ніж використання рівності (3.28). RSS знаходиться мінімізацією по $\boldsymbol{z}$ суми $\sum_i \sum_r (y_{ir} - z_i)^2$. Беручи похідні по z_i та прирівнюючи їх 0, одержимо

$$\hat{z}_i = \frac{\sum_r y_{ir}}{n_i} = \bar{y}_{i\cdot}, \; RSS = \sum_i \sum_r (y_{ir} - \bar{y}_{i\cdot})^2.$$

Для знаходження RSS_H можна обчислити похідні по $\beta_0, \ldots, \beta_{p-1}$ від виразу

$$\sum_i \sum_r (y_{ir} - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_{p-1} x_{p-1})^2$$

але простіше визначити вектор оцінок коефіцієнтів $\boldsymbol{b} = (b_0, \ldots b_{p-1})$, просто скориставшись виразом (3.16), врахувавши, що компонентами m-вимірного вектора $\overline{\boldsymbol{Y}}$ служать величини $\bar{y}_{i\cdot}$, $i = 1, \ldots, m$. В результаті одержимо

$$RSS_H = \sum_i \sum_r (y_{ir} - b_0 - b_1 x_{i1} - \ldots - b_{p-1} x_{i,\,p-1})^2 .$$

Для остаточної перевірки адекватності слід підставити знайдені вирази RSS і RSS_H у рівність (3.40) і порівняти знайдене значення з квантилем розподілу Фішера $F_{m-p,\,n-m}$ у відповідності з правилом перевірки ЗЛМ.

Відзначимо ще один вираз для F- відношення (3.40), для чого наведемо зараз рівність, що досить часто зустрічається в різних питаннях лінійного регресійного аналізу [33, 35]:

$$\sum_{i=1}^{m} \sum_{r=1}^{n_i} (y_{ir} - \hat{y}_i)^2 = \sum_{i=1}^{m} \sum_{r=1}^{n_i} (y_{ir} - \bar{y}_{i\cdot})^2 + \sum_{i=1}^{m} n_i (\hat{y}_i - \bar{y}_{i\cdot})^2 , \qquad (3.41)$$

де $\hat{y}$ — оцінка функції регресії (8.15), $\hat{y}_i = \hat{y}(\boldsymbol{x}^i)$, $i = 1, \ldots, m$.

З урахуванням (3.41), F- статистика (3.40) приймає вигляд

$$F = \frac{n-m}{m-p} \cdot \frac{\sum_{i=1}^{m} n_i (\hat{y}_i - \bar{y}_{i\cdot})^2}{\sum_{i=1}^{m} \sum_{r=1}^{n_i} (y_{ir} - \bar{y}_{i\cdot})^2} .$$

Одержаний вираз співпадає з відношенням S_1^2 / S_2^2 (визначення останнього — після рівності (3.15)), і тим самим вищезазначений спосіб перевірки

адекватності моделі (3.31), що заснований на обчисленні даного відношення, одержує додаткове обґрунтування.

2.6. Відбір «найкращої» регресії

4.1.1 Вступ. Однією з основних задач регресійного аналізу є розв'язання питання, які саме регресори або предикторні змінні слід включати в модель. Нехай $x_1, \dots x_p$ — повний набір всіх можливих регресорів, який містить такі функції як квадрати, мішані добутки і взагалі всі функції, що здаються доцільними. Для відбору деякої підмножини з цієї сукупності регресорів є дві протилежні за своїм характером можливості. З одного боку, якщо ми хочемо одержувати надійні прогнози на базі обраної моделі, нам треба включити якомога більше пояснюючих змінних. З іншого боку, враховуючи витрати, що пов'язані із включенням в модель надто великої кількості регресорів, бажано включати їх якомога менше. Цю ситуацію часто характеризують як потребу вибрати найкращу регресію. Слід мати на увазі, що термін «найкраща» є у даній ситуації суб'єктивним. Не існує ніякої єдиної статистичної процедури для відбору відповідної кількості параметрів регресії, і кожний метод носить елементи суб'єктивності.

В даній лекції буде розглянуто метод відбору регресорів, що спирається на поступове розширення початкової множини пояснюючих змінних, причому на кожному кроці доцільність розширення зазначеної множини перевіряється з допомогою так званого метода найменшої додаткової суми квадратів [3] .

4.1.2. **Покрокове додавання регресорів**. Припустимо, що вже після того, як підібрана модель регресії

$$MY = X\beta,\ DY = \sigma^2 I_n,$$

де M та D — символи математичного сподівання і дисперсії відповідно, I_n— одинична матриця розміру $n \times n$:

$$MY = \begin{pmatrix} My_1 \\ \dots \\ My_n \end{pmatrix}, DY = \begin{pmatrix} Dy_1 & cov(y_1, y_2) & \dots & cov(y_1, y_n) \\ cov(y_2, y_1) & Dy_2 & \vdots\vdots & cov(y_2, y_n) \\ cov(y_n, y_1) & cov(y_n, y_2) & \dots & Dy_n \end{pmatrix},$$

ми хочемо включити в неї додаткові регресори x_j, щоб модель з введенням цих факторів прийняла вигляд

$$G: MY = X\beta + Z\gamma = (X, Z)\begin{pmatrix} \beta \\ \gamma \end{pmatrix} = W\delta\ . \tag{4.1}$$

Тут позначено $(X, Z) = W, \gamma = \begin{pmatrix} \beta_p \\ \beta_{p+1} \\ \dots \\ \beta_q \end{pmatrix}$ і $\begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \delta$, де X — матриця розміру $n \times p$ рангу p, Z — матриця розміру $n \times q$ рангу q:

$$X = \begin{pmatrix} x_{10} x_{11} \dots x_{1,p-1} \\ x_{20} x_{21} \vdots\vdots x_{2,p-1} \\ \dots \dots \vdots\vdots \dots \\ x_{n0} x_{n1} \cdots x_{n,p-1} \end{pmatrix}, Z = \begin{pmatrix} x_{1,p} x_{1,p+1} \dots x_{1,p+q} \\ x_{2,p} x_{2,P+1} \vdots\vdots x_{2,p+q} \\ \dots \dots \vdots\vdots \dots \\ x_{n,p} x_{n,p+1} \cdots x_{n,p+q} \end{pmatrix}.$$

Припускається, що стовпці матриці Z лінійно не залежать від стовпців матриці X , тобто матриця $W = (X, Z)$ розміру $n \times (q + p)$ має ранг $q + p$. Тоді маємо дві можливості знаходження оцінки найменших квадратів $\hat{\delta}_G$Вектора δ. По-перше, можемо знайти $\hat{\delta}_G$і її дисперсійну матрицю безпосередньо із співвідношень

$$\hat{\delta}_G = (W'W)^{-1}W'Y, D\hat{\delta}_G = \sigma^2(W'W)^{-1}.$$

По-друге, можна зменшити кількість необхідних викладок, якщо використати обчислення, що були зроблені в процесі підбору моделі. Відповідні результати сформульовано у наступній теоремі 9.1.

4.1.3. Теорема. Нехай $R = I_n - X(X'X)^{-1}X'$, $R_G = I_n - W(W'W)^{-1}W', L = (X'X)^{-1}X'Z, M = (Z'RZ)^{-1}$ $\hat{\delta}_G = \begin{pmatrix}\hat{\beta}_G \\ \hat{\gamma}_G\end{pmatrix}$.

Тоді

$$\hat{\beta}_G = (X'X)^{-1}X'(Y - Z\hat{\gamma}_G) = \hat{\beta} - L\hat{\gamma}_G. \tag{4.2}$$

$$\hat{\gamma}_G = (Z'RZ)^{-1}Z'Z'RZ. \tag{4.3}$$

$$Y'R_GY = (Y - Z\hat{\gamma}_G)'R(Y - Z\hat{\gamma}_G). \tag{4.4}$$

$$Y'R_GY = Y'RY - \hat{\gamma}_GZ'RY. \tag{4.5}$$

$$D\hat{\delta}_G = \sigma^2\begin{pmatrix}(X'X)^{-1} + LML', & -LM \\ -ML', & M\end{pmatrix}. \tag{4.6}$$

Пояснимо, що рівність (4.2) дає вектор коефіцієнтів при змінних x_0, $x_1,\ldots, x_{p-1}$ в розширеній моделі G, (4.3) — аналогічний вектор для x_p, x_{p+1} ,…, x_{p+q}, (4.4) і (4.5) — остаточну суму квадратів розширеної моделі і, нарешті, (4.6) — вираз для дисперсійної матриці оцінки $\widehat{\delta_G}g$ вектора коефіцієнтів при регресорах в розширеній моделі.

4.1.4 Метод додаткової суми квадратів. Як вже відзначалося у вступі, в регресійних задачах часто постає питання, чи має сенс включати в модель ті або інші члени. Відповідь на нього можна одержати, припускаючи, що ці члени вже включені в модель, і досліджуючи виникаючий при цьому додатковий вклад в суму квадратів, що зумовлена регресією. Середній

квадрат, що одержаний з цієї *додаткової суми*, можна потім порівняти з оцінкою s^2 дисперсії σ^2. Якщо середній квадрат значимим чином перевищує оцінку σ^2, то має сенс включити в модель нові члени. В протилежному випадку їх включення недоцільне.

Такий приклад наведено, наприклад, в [3]: при підборі прямої лінії, коли величина СК (b_1/b_0) являв собою додаткову суму квадратів, яка зумовлена включенням в модель члена $\beta_1 X$. Нижче ми розглянемо більш загальну процедуру. Нехай $Z_1, Z_2, \dots, Z_{p-1}$ — відомі функції основних змінних $X_1, X_2, \dots,$.Припустимо, що значення предикторів і відповідного відгуку Y нам відомі. Розглянемо дві моделі:

Модель 1:

$$Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_{p-1} Z_{p-1} + \varepsilon.$$

Допустимо, що одержано наступні МНК- оцінки: $b_0(1), b_1(1), b_2(1), \dots, b_{p-1}(1)$, і СК$(b_0(1), b_1(1), b_2(1), \dots, b_{p-1}(1)) = S_1$, причому модель є адекватною. Крім того, нехай s^2— оцінка σ^2, що одержана з залишків моделі 1.

Модель 2: $Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_{q-1} Z_{q-1} + \varepsilon, q < p$.

Величини Z в моделі 2 — ті ж самі функції, що й в моделі 1, з тими ж самими нижніми індексами. Проте їх менше, ніж в моделі 1. Припустимо далі, що ми одержали наступні МНК-оцінки

$$b_0(2), b_1(1), b_2(2), \dots, b_{q-1}(2).$$

Вони можуть співпадати або не співпадати з оцінками $b_0(1), b_1(1), b_2(1), \dots, b_{p-1}(1)$. Якщо вони співпадають, $b_i(1)$ і $b_j(1)$ є ортогональними лінійними функціями при $1 \le i \le q-1, q \le j \le p-1$.

Останнє має місце, коли в моделі 1 всі перші q стовпців матриці $\boldsymbol{X}$ ортогональні останнім $p-q$ стовпцям. Така ситуація, як правило, виникає у випадку спланованого експерименту. В протилежному випадку це малоймовірно.

Для другої моделі ми покладемо СК$(b_0(2), b_1(2), b_2(2), \dots, b_{q-1}(2)) = S_2$. Тоді різниця $S_1 - S_2$ являє собою додаткову суму квадратів (ДСК), яка зумовлена включенням членів $\beta_q Z_q + \beta_{q+1} Z_{q+1} + + \cdots + \beta_{p-1} Z_{p-1}$ в модель 1. Оскільки S_1 має p степенів свободи, а S_2 — q степенів свободи, величина $S_1 - S_2$ має p – q степенів свободи. Можна показати [11,19] , що коли $\beta_q = \beta_{q+1} = \cdots = \beta_{p-1} = 0$, то $M\frac{S_1-S_2}{p-q} = \sigma^2$. Крім того, якщо помилки розподілені за нормальним законом, то величина $S_1 - S_2$ буде розподілена як $\sigma^2\chi^2_{p-q}$ незалежно від s^2. Це означає, що ми можемо порівняти величину $\frac{S_1-S_2}{p-q}$ з s^2 за допомогою $F(p-q, \nu)$-критерія, де ν — кількість степенів свободи, на якій заснована оцінка s^2, і застосувати цю процедуру для перевірки гіпотези $H_0: \beta_q = \beta_{q+2} = \cdots = \beta_{p-1} = 0$.

Для $S_1 - S_2$ часто застосовується позначення СК$(b_q, \dots, b_{p-1} | b_0, \dots, b_{q-1})$, яке розшифровується як сума квадратів $b_q, \dots, b_{p-1}$ при заданих $b_0, \dots, b_{q-1}$. Проте слід мати на увазі, що насправді тут розглядаються дві моделі, хоча з позначень це не очевидно. За тим же принципом ми можемо послідовно одержати для довільної регресійної моделі СК(b_0), СК$(b_1|b_0)$,…, СК$(b_{p-1}|b_0, \dots, b_{p-2})$. Всі ці суми квадратів розподілені незалежно від s^2 і дорівнюють своїм середнім квадратам, оскільки мають по одній степені свободи. Середні квадрати можуть бути співставленні з величиною s^2 за допомогою послідовності *F*- критеріїв. Так перевірки особливо доцільні, коли

члени моделі мають логічний характер входження , наприклад, як в ситуації $Z_j = X^j$. Тоді можна зробити висновок, скільки членів має бути у моделі.

Поліноміальні моделі. Перевірка гіпотез і інші питання

Якщо члени, котрі містяться в моделі, згруповано натуральним чином, як, наприклад, в *поліноміальних моделях*, що містять а) вільний член β_0; б) члени першого порядку; в) члени другого порядку і т.д., то ми можемо будувати різні ДСК, наприклад СК(оцінки b_iчленів першого порядку $|b_0$), СК(оцінки b_{ij} членів другого порядку$|b_0$,оцінки b_i членів першого порядку) і порівняти їх з величиною s^2. ПДК можна застосовувати по-різному з тим, щоб одержати такий розклад суми квадратів, яке буле здаватись прийнятним для проблеми, що розглядається.

Число степенів свободи для кожної ДСК буле дорівнювати числу параметрів, що вказані в дужках перед вертикальною лінією. Це буде саме так, якщо виключити випадок, коли оцінки лінійно залежні, що має місце тоді, коли матриця $\boldsymbol{X}$ має неповний ранг і нормальні рівняння залежні. Число степенів свободи тоді дорівнює максимальному числу лінійно незалежних оцінок в множині оцінок, що розглядаються. Припустимо, що цей виняток не має місця. ДСК розподілені незалежно від s^2. Поділивши на s^2 відповідний середній квадрат, одержимо F-відношення для перевірки гіпотези, що справжнє значення коефіцієнта, включенням якого зумовлена поява даного ДСК, дорівнює 0. Принцип додаткової суми квадратів фактично є частинним випадком перевірки загальної лінійної гіпотези (ЗЛГ). При більш загальному підході додаткова сума квадратів обчислюється з залишкових сум квадратів, а не сум квадратів, що зумовлена регресією. Оскільки в обох випадках використовується спільна сума квадратів $\boldsymbol{Y'\ Y}$, то і результати обчислень

будуть спільними.

Наведемо більш формальне викладення вищесказаного, перейшовши до дещо інших позначень. Нехай є дві моделі:

$$y^{(1)} = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{q-1} x_{q-1} \dots + \beta_{p-1} x_{p-1} + \varepsilon, \quad (4.7)$$

$$y^{(2)} = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{q-1} x_{q-1} + \varepsilon, \;\; q < p \quad (4.8)$$

Перевіряється та точка зору, що змінні $x_q, \dots, x_{p-1}$ є зайвими, чому відповідає гіпотеза $H_0: \beta_q = \dots = \beta_{p-1} = 0$. Нехай відповідно до двох вказаних моделей маємо дві матриці спостережень регресорів $\boldsymbol{X}_1, \boldsymbol{X}_2$ і вектор спостережень відгуку $\boldsymbol{Y}$. Нехай розмірності матриць $\boldsymbol{X}_1, \boldsymbol{X}_2$ будуть $n \times p$ і $n \times q$ відповідно ($n > max\{p, q\}$). Обчислимо оцінки коефіцієнтів регресії для моделей (5.1), (5.2). Позначимо $\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}$ вектори зазначених оцінок:

$$\boldsymbol{b}^{(1)} = \begin{pmatrix} b_0^1 \\ b_1^1 \\ \dots \\ b_{q-1}^1 \\ b_q^1 \\ \dots \\ b_{p-1}^1 \end{pmatrix}, \boldsymbol{b}^{(2)} = \begin{pmatrix} b_0^2 \\ b_1^2 \\ \dots \\ b_{q-1}^2 \end{pmatrix}.$$

Рівності для знаходження $\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}$ наступні:

$$\boldsymbol{b}^{(1)} = (\boldsymbol{X}_1^t \boldsymbol{X}_1)^{-1} \boldsymbol{X}_1^t \boldsymbol{Y}\,, \quad (4.9).$$

$$\boldsymbol{b}^{(2)} = (\boldsymbol{X}_2^t \boldsymbol{X}_2)^{-1} \boldsymbol{X}_2^t \boldsymbol{Y}, \quad (4.10).$$

де верхній індекс *t* є символом транспонування матриці. Позначимо також $\boldsymbol{x}_i = \left(x_{i0}, x_{i1}, \dots, x_{i,q-1,}, x_{i,q}, x_{i,p-1}\right)$ $(i = 1, \dots, n)$ *i*-й рядок об'єднаної матриці предикторів, y_i — відповідне йому *i*-те значення змінної відгуку *y*,

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

Далі для кожного $i = 1, \dots, n$ обчислимо

$$y_i^1 = b_0^1 x_{i0} + b_1^1 x_{i1} + \cdots + b_{q-1}^1 x_{i,q-1} + b_q^1 x_{i,q} + \cdots + b_{p-1}^1 x_{i,p-1},$$

$$y_i^2 = b_0^2 x_{i0} + b_1^2 x_{i1} + \cdots + b_{q-1}^2 x_{i,q-1}.$$

Знайдемо тепер суми

$$S_1^2 = \sum_{i=1}^{n} (y_i^1 - \bar{y})^2, S_2^2 = \sum_{i=1}^{n} \left(y_i^2 - \bar{y}\right)^2, S^2 = \frac{1}{n-p}\sum_{i=1}^{n} (y_i - y_i^1)^2.$$

На базі обчислених сум складемо *F*- відношення

$$F = \frac{(S_1^2 - S_2^2)/(p-q)}{S^2}. \tag{4.11}$$

Зауважимо, що чисельник у виразі (4.11) є невід'ємною величиною. Дійсно, маємо рівність

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (y_i - y_i^k)^2 + \sum_{i=1}^{n} (y_i^k - \bar{y})^2, k = 1,2.$$

Остання рівність — наслідок так званої основної тотожності дисперсійного аналізу [3,4]. Звідси оскільки з того, що $p > q$ випливає, що

$$\sum_{i=1}^{n} (y_i - y_i^1)^2 \le \sum_{i=1}^{n} (y_i - y_i^2)^2,$$

то маємо

$$\sum_{i=1}^{n}(y_i^1 - \bar{y})^2 \geq \sum_{i=1}^{n}(y_i^2 - \bar{y})^2.$$

Як доведено, наприклад, в [3,4], за умови справедливості гіпотези H_0 F-відношення (5.5) має розподіл Фішера $F(p-q, n-p)$. Звідки випливає правило перевірки H_0: якщо відношення (5.5) перевищує квантиль рівня $1-\alpha$ зазначеного розподілу, то гіпотеза H_0 відхиляється при імовірності помилки 1-го роду, рівній вказаній величині α.

Альтернативні представлення

Можна вирішити, чи треба видаляти коректувальний фактор до того, як визначиться різниця між сумами квадратів для одержання додаткової суми квадратів. Наприклад, припустимо, що початкова модель має вигляд

$$y^{(1)} = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \varepsilon, \qquad (4.12).$$

а «вкорочена» модель наступна:

$$y^{(2)} = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon\ . \qquad (4.13).$$

Для моделі (1) сума квадратів, що зумовлена регресією, має вигляд

$$\text{СК}\ (b_0, b_1, b_2, b_3, b_4, b_5) = S_1^2, \qquad (4.14)$$

корегуючий фактор дорівнює $n\bar{y}^2$ а загальна сума квадратів — $\boldsymbol{Y'Y}$. Отже, остаточна сума квадратів дорівнює $\boldsymbol{Y'}\ \boldsymbol{Y} - S_1^2$. Для моделі 2 сума квадратів, що зумовлена регресією, має вигляд

$$\text{СК}(b_0, b_1, b_2) = S_2^2. \qquad (4.15)$$

Остаточна сума квадратів дорівнює $\boldsymbol{Y'Y} - S_2^2$. Припустимо, що нам потрібна наступна сума квадратів:

$\text{СК}(b_3, b_4, b_5|b_0, b_1, b_2) = S_1^2 - S_2^2$.

Ми можемо переписати цей вираз в альтернативному вигляді

$S_1^2 - S_2^2 = (S_1^2 - n\bar{y}^2) - (S_2^2 - n\bar{y}^2)$,

і тоді він перетворюється на різницю між сумами квадратів, скорегованих на b_0, тобто

$\text{СК}(b_1, b_2, b_3, b_4, b_5|b_0) - \text{СК}(b_1, b_2|b_0)$.

Є ще і третій спосіб одержання додаткової суми квадратів. Можна представити різницю $S_1^2 - S_2^2$ як

$S_1^2 - S_2^2 = (\boldsymbol{Y}'\boldsymbol{Y} - S_2^2) - (\boldsymbol{Y}'\boldsymbol{Y} - S_1^2)$.

У такому разі вона перетворюється у зворотну різницю залишкових сум квадратів, оскільки регресійна модель з більшою сумою квадратів, що зумовлена регресією, повинна мати меншу остаточну суму квадратів і навпаки.

2.7. **Введення в дисперсійний аналіз**

В реальних ситуаціях, що виникають при аналізі статистичних даних, пов'язаних з проблемами промисловості, економіки, наукових досліджень, часто зустрічаються випадки, коли важливі фактори, що впливають на досліджувану величину, процес або явище, носять *якісний* характер. Тоді сукупність статистичних методів, що забезпечують оцінки параметрів математичних моделей, які описують досліджувану залежність та перевірку різноманітних гіпотез відносно згаданих моделей, носить назву *дисперсійного*

аналізу (ДА або англійський варіант — ANOVA). Якщо кількість вказаних факторів дорівнює 2, то мова йде про *двофакторний дисперсійний* аналіз.

Моделі дисперсійного аналізу можуть включати *сталі* (*постійні*) *ефекти*, *випадкові ефекти* або комбінації таких ефектів [2,4].

Зауважимо, що у випадку нормальної розподіленості залежної змінної перевірка статистичних гіпотез для моделей ДА базується на обчисленні так званих *F*- відношень Фішера і порівнянням обчислених значень з критичними точками, що являють собою квантилі розподілів Фішера з певними (цілими) кількостями степенів свободи [2,4]. Ми будемо називати такий спосіб перевірки класичним. Зауважимо, що коли нормальна розподіленість не має місця або принаймні є сумнівною, то існує так званий робастний спосіб перевірки таких гіпотез, що також ґрунтується на розподілах Фішера, але з іншими, дробовими степенями свободи [7].

2.7.1 Класифікація за однією ознакою

Вище було показано, як загальну теорію регресії можна застосувати до задачі порівняння двох нормальних сукупностей у випадку, коли дисперсії цих двох сукупностей рівні між собою. Узагальнимо цю теорію на випадок порівняння I нормальних сукупностей для $I \geq 2$, дотримуючись схеми, що застосовується в [4].

Нехай y_{ij} — значення *j*-го спостереження (j = 1,2,…,J) над i –ю нормальною сукупністю $N(\mu_i, \sigma)(i = 1, \dots, I)$. Позначимо $y_{i\cdot}$ вибіркове середнє значення, що відповідає *i*-й сукупності. Застосуємо відповідні результати загального регресійного аналізу до розв'язання поставленої задачі. З цією метою представимо нашу інформацію у наступному вигляді:

$$y_{ij} = \mu_i + \varepsilon_{ij}, i = 1, \dots, I, j = 1, \dots J,$$

де ε_{ij} — незалежні випадкові величини з розподілом $N(0, \sigma)$. Для зведення поставленого питання до перевірки деякої лінійної гіпотези, використаємо векторно-матричний запис наших даних. Одержимо наступне:

$$\begin{pmatrix} y_{11} \\ y_{12} \\ \dots \\ y_{1J} \\ y_{21} \\ y_{22} \\ \dots \\ y_{2J} \\ \dots \\ y_{I1} \\ y_{I2} \\ \dots \\ y_{IJ} \end{pmatrix} = \begin{pmatrix} 100 \dots 0 \\ 100 \dots 0 \\ \dots \dots \dots \\ 10 \dots 0 \\ 010 \dots 0 \\ 010 \dots 0 \\ \dots \dots \dots \\ 010 \dots 0 \\ \dots \dots \dots \\ 000 \dots 1 \\ 000 \dots 1 \\ \dots \dots \dots \\ 000 \dots 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_I \end{pmatrix} + \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{12} \\ \dots \\ \varepsilon_{1J} \\ \varepsilon_{21} \\ \varepsilon_{22} \\ \dots \\ \varepsilon_{2J} \\ \dots \\ \varepsilon_{I1} \\ \varepsilon_{I2} \\ \dots \\ \varepsilon_{IJ} \end{pmatrix} \quad (5.1)$$

або

$$\boldsymbol{Y} = \boldsymbol{X\mu} + \boldsymbol{E}, \quad (5.2)$$

де літери, набрані напівжирним шрифтом, є очевидними скороченими позначеннями для векторів і матриць з рівності (5.1). Оскільки стовпці матриці $\boldsymbol{X}$ лінійно незалежні, то (5.2) являє собою частинний випадок лінійної регресії з матрицею плану повного рангу (див. рівність (3.4)). Гіпотеза, що нас цікавить, має вигляд $H: \mu_1 = \mu_2 = \dots = \mu_I$

або $\mu_1 - \mu_I = \mu_2 - \mu_I = \dots = \mu_{I-1} - \mu_I = 0.$

В матричній формі це виглядає наступним чином:

$$\begin{pmatrix} 100,,,0-1 \\ 010\ldots0-1 \\ \ldots\ldots\ldots\ldots\ldots \\ 000\ldots1-1 \end{pmatrix}\begin{pmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_I \end{pmatrix} = \mathbf{0}. \qquad (5.3)$$

Матриця в лівій частині останньої рівності має I-1 рядок і I стовпців. Її рядки очевидним чином лінійно незалежні, тому її ранг дорівнює I-1. Бачимо, що наша гіпотеза H відноситься до класу лінійних гіпотез, і до її перевірки можна застосувати теорію, що викладена вище в рамках регресійного аналізу. В результаті такого застосування маємо [4] статистику критерію

$$F = \frac{J\sum_i(\bar{y}_{i\cdot}-\bar{y}_{\cdot\cdot})^2/(I-1)}{\sum_{i,j}\left(y_{ij}-\bar{y}_{i\cdot}\right)^2/(IJ-1)}, \qquad (5.4)$$

де $\bar{y}_{i\cdot}$ — середнє по і-й сукупності, $\bar{y}_{\cdot\cdot}$ — загальне середнє. Якщо гіпотеза H є вірною, то ця статистика має розподіл Фішера $F_{I-1,IJ-1}$.

Помітимо, що модель (5.1) може бути представлена у вигляді

$$Z_r = \mu_1 d_{r1} + \mu_2 d_{r2} + \cdots + \mu_I d_{rI} + \varepsilon_r, r = 1,2,\ldots,n \qquad (5.5)$$

де

$$n = IJ\ . \qquad (5.6)$$

Таке представлення знаходить застосування в питаннях слушних перепараметризацій моделі [4].

Для швидких обчислень корисні також наступні співвідношення:

$$\sum_{i,j}\left(\bar{y}_{i\cdot}-\bar{y}_{\cdot\cdot}\right)^2 = \sum_i \frac{(y_{i\cdot})^2}{J} - \frac{(y_{\cdot\cdot})^2}{IJ} \qquad (5.7)$$

де

$$y_{i\cdot} = \sum_j y_{ij}, \quad y_{\cdot\cdot} = \sum_{i,j} y_{ij} \qquad (5.8)$$

та

$$\sum_{i,j}(y_{ij} - \overline{y}_{i.})^2 = \sum_{i,j} y_{ij}^2 - \sum_j \frac{y_{i.}^2}{J} \tag{5.9}$$

Нерівні числа спостережень на кожне середнє

Якщо ми маємо J_i спостережень на кожну *i*-ту нормальну сукупність, то у даному разі для величини величина *n* маємо співвідношення

$$n = \sum_{i=1}^{I} J_i \tag{5.10}$$

Відповідною *F*-статистикою для перевірки гіпотези *H* буде вираз

$$F = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J_i}(\overline{y}_{i.} - \overline{y}_{..})^2/(I - 10}{\sum_{i=1}^{I}\sum_{j=1}^{J_i}(y_{ij} - \overline{y}_{i.})^2/(\sum_i J_i)} \tag{5.11}$$

2.7.2. Двостороння класифікація промислових експериментів (двофакторний дисперсійний аналіз)

В цьому підрозділі розглядаються моделі, що використовуються для аналізу диференційних ефектів при двох факторах *A*, *B*. Кожний фактор, за означенням, має скінченну кількість рівнів:

$$A \text{ — } 1,2,\ldots,I,\ B \text{ — } 1,2,..,J. \tag{5.12}$$

Визначимо два типи відношень між факторами, що називаються *перетином* і *групуванням*. Два фактори *A* і *B* називаються такими, що перетинаються (це позначається $A \times B$), якщо в плані представлено всі можливі сполучення рівнів факторів. Комбінацію (*i*, *j*), де *i* позначає рівень фактора *A*, а *j* — рівень фактора *B*, часто називають *i j*- чарункою, $i = 1,\ldots,I$, *j*

= 1,…,*J*. В кожній чарунці розглядається значення випадкової величини *Y* на K_{ij} вибіркових експериментальних одиницях. Цю ситуацію можна уявити і по - іншому. Кожній (*i j*)- чарунці призначається єдина експериментальна одиниця, і K_{ij} разів вимірюється значення випадкової величини *Y* . У кожному разі *ij*- чарунці відповідає випадкова вибірка

$$y_{ij1}, y_{ij2},\ldots, y_{ijK_{ij}}, i = 1,\ldots,I, j = 1,\ldots,J. \quad (5.13)$$

Модель з двома факторами, що перетинаються, називають ***двофакторним планом***, ***двофакторною класифікацією*** або ***факторною моделлю з двома факторами*** [2,4].

Інколи зустрічається і інший вид відношень між факторами. Кажуть, що фактор *B* групує фактор *A*, якщо кожний рівень фактора *B* зустрічається в парі не більш ніж з одним рівнем фактору *A*. Таке відношення позначається *B*(*A*). Кажуть, що фактор *A* групує фактор *B*, що *A* — фактор, що групує, а фактор *B* — згрупований. У цьому випадку загальна кількість комбінацій, при яких виконуються виміри, менше, ніж *IJ*.

Крім того, має сенс розрізняти дві ситуації: фіксовані ефекти — модель I і випадкові ефекти — модель II.

Розглянемо спочатку випадок відношення $A \times B$ і модель I для ситуації, коли для кожної комбінації факторів можливо провести повторні спостереження (в однаковій кількості).

2.7.2.1. Двофакторні плани з повтореннями спостережень

Надалі будемо вважати, задані фактор *A* з *I* рівнями і фактор *B* з *J* рівнями. Експерименти, що відповідають всім можливим комбінаціям рівнів,

повторюються одне й те ж саме число $K > 1$ разів. Нехай y_{ijk} позначає значення змінної Y, що одержано при k- му повторенні експерименту в чарунці (i,j),

$$i = 1,\ldots,I, j = 1,\ldots,J, k = 1,\ldots, K . \quad (5.14)$$

Якщо обидва фактори відповідають моделі I, то двофакторний план з фіксованими ефектами задається співвідношенням

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}, \quad (5.15)$$

$i = 1,\ldots,I, j = 1,\ldots,J, k = 1,\ldots, K.$

Тут μ — загальне або генеральне середнє, α_i — ефект рядка або i-й диференціальний або головний ефект фактору A, β_j — ефект стовпця або j-й диференціальний ефект фактору B. Величина $(\alpha\beta)_{ij}$ називається (двофакторною) ***взаємодією*** i-го рівня фактору A і j-го рівня фактору B. Ця величина враховує диференціальний ефект комбінацій i-го рівня фактору A і j-го рівня фактору B, якщо він не виражається сумою $\mu + \alpha_i + \beta_j$. Модель, в якій взаємодія $(\alpha\beta)_{ij}$ дорівнює $0\ \forall\ i, j$, називається ***адитивною***. Нарешті, помилки ε_{ijk}, за припущенням, є незалежними і нормально розподіленими з нульовим середнім значенням і деякою дисперсією σ^2.

За прийняттям моделі (5.15) першою задачею є оцінка її коефіцієнтів (параметрів) μ, α_i, β_j, $(\alpha\beta)_{ij}$ та дисперсії величини ε_{ijk}. Для даної моделі оцінки МНК (найменших квадратів) неоднозначні, тому на диференціальні ефекти накладаються додаткові обмеження [2,4,17]:

$$\sum_{i=1}^{I}\alpha_i = 0\,,\ \sum_{j=1}^{J}\beta_j = 0,$$

$$\sum_{i=1}^{I}(\alpha\beta)_{ij}=0\ \forall\, j,\ \sum_{j=1}^{J}(\alpha\beta)_{ij}=0\quad \forall i. \tag{5.16}$$

При цих обмеженнях МНК-оцінки стають однозначними і при цьому одержуємо: величини

$$\hat{\mu}=\bar{y}_{...} \tag{5.17}$$

$$\hat{\alpha}_i=\bar{y}_{i..}-\bar{y}_{...}, \tag{5.18}$$

$$\hat{\beta}_j=\bar{y}_{.j.}-\bar{y}_{...}, \tag{5.19}$$

$$\hat{(\alpha\beta)}_{ij}=\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y}_{...}, \tag{5.20}$$

де

$$\bar{y}_{i..}=\frac{1}{KJ}\sum_{j=1}^{J}\sum_{k=1}^{K}y_{ijk}\ , \tag{5.20}$$

$$\bar{y}_{.j.}=\frac{1}{KI}\sum_{i=1}^{I}\sum_{k=1}^{K}y_{ijk}\ , \tag{5.21}$$

$$\bar{y}_{ij.}=\frac{1}{K}\sum_{k=1}^{K}y_{ijk}\ , \tag{5.22}$$

$$\bar{y}_{...}=\frac{1}{IJK}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}y_{ijk}\ , \tag{5.23}$$

є незміщеними оцінками, відповідно, величин μ, α_i, β_j, $(\alpha\beta)_{ij}$. Зауважимо, що величини $\bar{y}_{i..}$, $\bar{y}_{.j.}$, $\bar{y}_{ij.}$, $\bar{y}_{...}$ — це, відповідно, середнє по всім спостереженням у рядку i, середнє по всім спостереженням у стовпці j, середнє по всім спостереженням у точці (i, j), середнє по всім спостереженням. Незміщеною оцінкою дисперсії (σ^2) випадкової складової ε_{ijk} є величина Q_R з наведеної нижче таблиці 1.2.3.1.

2.7.2.2 Основні гіпотези двофакторної класифікації і таблиця двофакторного дисперсійного аналізу

Важливу роль у кінцевому з'ясуванні вигляду моделі (1) відіграють наступні гіпотези, що позначені нижче як $\boldsymbol{H_{AB}}$, $\boldsymbol{H_A}$, і $\boldsymbol{H_B}$.

$\boldsymbol{H_{AB}}$: всі $(\alpha\beta)_{ij}$ дорівнюють **0** .

якщо ця гіпотеза підтверджується, то це означає, що взаємодія факторів A і B відсутня. Тоді в моделі (5.1) можна вважати, що всі $(\alpha\beta)_{ij}$ дорівнюють 0.

Якщо гіпотеза $\boldsymbol{H_{AB}}$ приймається, то переходимо до перевірки гіпотез $\boldsymbol{H_A}$ і $\boldsymbol{H_B}$.

$\boldsymbol{H_A}$: всі α_i дорівнюють 0, тобто ефекти рядків відсутні;

$\boldsymbol{H_B}$: всі β_j дорівнюють 0, тобто ефекти стовпців відсутні.

Відзначимо, що за прийняттям гіпотези $\boldsymbol{H_{AB}}$ гіпотези $\boldsymbol{H_A}$ і $\boldsymbol{H_B}$ втрачають зміст і перевірці не підлягають. Статистики для перевірки зазначених гіпотез будуються на основі величин з останнього стовпця таблиці 2.5.1

Таблиця 2.7.1 **Таблиця дисперсійного аналізу з повторенням спостережень при класифікації за двома ознаками**.

Джерело дисперсії	Сума квадратів	Степені свободи	Середній квадрат
Фактор A	$SS_A = JK\sum_{i=1}^{I}\hat{\alpha}_i^2$	$\nu_A = I - 1$	$Q_A = SS_A / \nu_A$
Фактор B	$SS_B = IK\sum_{j=1}^{J}\hat{\beta}_j^2$	$\nu_B = J - 1$	$Q_B = SS_B / \nu_B$
Взаємодія	$SS_{AB} = K\sum_{i=1}^{I}\sum_{j=1}^{J}(\alpha\beta)_{ij}^2$	$\nu_{AB} = (I - -1)(J - 1)$	$Q_{AB} = SS_{AB} / \nu_{AB}$
Залишок	$SS_R = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(y_{ijk} - \bar{y}_{ij.})^2$	$\nu_R = IJ(K - -1)$	$Q_R = SS_R / \nu_R$
Повна	$SS_T = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(y_{ijk} - \bar{y}_{...})^2$	$\nu_T = IJK - -1$	$Q_T = SS_T / \nu_T$

Зазначені статистики, що перевіряють гіпотези $\boldsymbol{H_{AB}}$, $\boldsymbol{H_A}$ і $\boldsymbol{H_B}$ є всі без винятку F- статистиками (тобто такими, що при справедливості відповідних гіпотез мають розподіли Фішера). Їх вирази з застосуванням очевидних позначень $\boldsymbol{F_{AB}}$, $\boldsymbol{F_A}$, $\boldsymbol{F_B}$ наводяться нижче.

$$\boldsymbol{H_{AB}}: \boldsymbol{F_{AB} = Q_{AB} / Q_R} \; ; \qquad (5.24)$$

$$\boldsymbol{H_A}: \boldsymbol{F_A} = \boldsymbol{Q_A} / \boldsymbol{Q_R}; \qquad \boldsymbol{H_B} : \boldsymbol{F_B} = \boldsymbol{Q_B} / \boldsymbol{Q_R}. \tag{5.25}$$

Перевірка вказаних гіпотез виконується за двома методами: **класичним** і так званим **робастним**. Класичне правило доцільне, коли величини ε_{ijk} мають **нормальний** розподіл імовірностей. Робастний метод застосовується, коли є сумніви відносно нормальної розподіленості величин ε_{ijk} [7] .

Згідно з класичним методом статистики $\boldsymbol{F_{AB}}$, $\boldsymbol{F_A}$, $\boldsymbol{F_B}$ при справедливості відповідних гіпотез $\boldsymbol{H_{AB}}$, $\boldsymbol{H_A}$ і $\boldsymbol{H_B}$ мають, відповідно наступні розподіли Фішера:

$$F((I-1)(J-1); IJ(K-1)), F(I-1; IJ(K-1)), F(J-1; IJ(K-1)). \tag{5.26}$$

Тому при рівні значущості α критерію перевірки (критичні множини правосторонні) критичні значення для вказаних гіпотез будуть, дорівнювати $u_{1-\alpha}$, де $u_{1-\alpha}$ — квантиль рівня $1-\alpha$ відповідного розподілу Фішера (з цілими степенями свободи, що дані вище в (5.26).

Зауваження. Зазначені дії по перевірці гіпотез ДА базуються на теоремах математичної статистики, що стосуються розподілів квадратичних форм від нормально розподілених випадкових величин. Наведемо тут відповідні відомості, скориставшись загальним формулюванням з [6].

Нехай $\boldsymbol{y}$ має невироджений багатовимірний нормальний розподіл, моменти якого задовольняють умові

$$M(\boldsymbol{y}) = \boldsymbol{\mu}, \; V(\boldsymbol{y}) = \boldsymbol{V},$$

і квадратична форма $\boldsymbol{Q} = \boldsymbol{Q}(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y}$ припускає розклад у суму квадратичних форм:

$$\boldsymbol{Q} = \boldsymbol{y}'\boldsymbol{A}\,\boldsymbol{y} = \sum_{i=1}^{k} \boldsymbol{y}'\boldsymbol{A}_i\,\boldsymbol{y} = \sum_{i=1}^{k} \boldsymbol{Q}_i,$$

де $\boldsymbol{A}_i$ деякі матриці, матриця $\boldsymbol{A}\,\boldsymbol{V}$ є ідемпотентною (матриця $\boldsymbol{X}$ називається ідемпотентною, якщо має місце рівність $\boldsymbol{X}^2 = \boldsymbol{X}$). Тоді матриця $\boldsymbol{Q}$ має розподіл $\chi^2(r, \boldsymbol{\mu}'\boldsymbol{A}\,\boldsymbol{\mu})$- розподіл (нецентральний χ^2 з параметрами r, $\boldsymbol{\mu}'\boldsymbol{A}\,\boldsymbol{\mu}$), де r дорівнює рангу матриці $\boldsymbol{A}$, і кожне з наведених нижче трьох умов веде за собою дві інших умови:

(а) $\sum_{i=1}^{r} r_i = r$, де r — ранг матриці $\boldsymbol{A}_i$;

(б) матриці $\boldsymbol{A}_i\,\boldsymbol{V}$ ідемпотентні, або, що те ж саме, форми $\boldsymbol{Q}_i$ мають розподіли $\chi^2(r_i, \boldsymbol{\mu}'\boldsymbol{A}_i\,\boldsymbol{\mu})$ відповідно;

(в) $\boldsymbol{A}_i\,\boldsymbol{V}\,\boldsymbol{A}_j = \boldsymbol{0}$ при $i \neq j$ ($\boldsymbol{0}$ — нуль-вектор), або, що те ж саме, форми

$\boldsymbol{Q}_i = \boldsymbol{Q}_i(\boldsymbol{y})$ є попарно незалежними.

2.7.3. Двофакторні плани з неповторними спостереженнями у моделі фіксованих ефектів

В даному підрозділі будемо вважати, що задано фактори A, B з I, J рівнями відповідно, причому в кожній чарунці величина Y спостережувана в точності 1 раз. Такий експеримент називатиметься неповторним (безповторним). В даній роботі будуються оцінки і знаходяться статистики для перевірки гіпотез у зазначеній моделі для випадку фіксованих ефектів.

Оскільки кількість спостережень K дорівнює 1, то оцінити ефект взаємодії $(\alpha\beta)_{ij}$ у моделі ДА

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}, \qquad (5.27)$$

$$i = 1,\ldots,I, j = 1,\ldots,J, k = 1,\ldots, K$$

виявляється неможливим [2,4,17,18]. Тому, якщо дані не суперечитимуть, в якості моделі ДА замість (5.27) береться співвідношення

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}, \tag{5.28}$$

$$i = 1,\ldots,I, j = 1,\ldots,J.$$

Інтерпретація: μ — загальне середнє, α_i — ефект рядка, β_j — ефект стовпця, ε_{ij} — випадкова складова. Для забезпечення єдиності МНК-оцінок накладаються додаткові умови

$$\sum_{i=1}^{I} \alpha_i = 0, \sum_{j=1}^{J} \beta_j = 0. \tag{5.29}$$

Введемо наступні оцінки:

$$\hat{\mu} = \bar{y}_{..} \tag{5.30}$$

$$\hat{\alpha}_i = \bar{y}_{i.} - \bar{y}_{..}, \tag{5.31}$$

$$\hat{\beta}_j = \bar{y}_{.j} - \bar{y}_{..}, \tag{5.32}$$

$$\widehat{(\alpha\beta)}_{ij} = y_{ij} - \bar{y}_{i.} - y_{.j} + \bar{y}_{..}, \tag{5.33}$$

де

$$\bar{y}_{i.} = \frac{1}{J}\sum_{j=1}^{J} y_{ij}, \tag{5.34}$$

$$\bar{y}_{\cdot j} = \frac{1}{I}\sum_{i=1}^{I} y_{ij}\ , \tag{5.35}$$

$$\bar{y}_{\cdot\cdot} = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij}\ . \tag{5.36}$$

Оцінки (5.30) —(5.32) є незміщеними оцінками, відповідно, величин μ, α_i, β_j. Зауважимо, що величини $\bar{y}_{i\cdot}$, $\bar{y}_{\cdot j}$, $\bar{y}_{\cdot\cdot}$ — це, відповідно, середнє по всім спостереженням у рядку i, середнє по всім спостереженням у стовпці j, середнє по всім спостереженням. Незміщеною оцінкою дисперсії (σ^2) випадкової складової ε_{ijk} є величина Q_R з наведеної нижче таблиці 1.2.3.1.

2.7.3.1. Основні гіпотези і таблиця двофакторного неповторного дисперсійного аналізу.

Ми бачили, що у випадку неповторного експерименту адитивна модель ДА (5.28) є вимушено менш загальною, ніж (5.27). Доцільність застосування адитивної моделі може бути і сумнівною. В такому разі рівність $(\alpha\beta) = 0$ (тобто гіпотеза про відсутність взаємодії) потребує перевірки.

Але оскільки в моделі (5.27) не всі коефіцієнти можуть бути оціненими, то гіпотезу $(\alpha\beta)_{ij} = 0$, проти загального класу альтернатив $(\alpha\beta)_{ij} \neq 0$, i=1,…, I, j = 1,…, J перевірити неможливо. Проте якщо припустити, що $(\alpha\beta)_{ij} = G\ \alpha_i\ \beta_j$, то для перевірки гіпотези $\{G = 0\}$ проти гіпотези $\{G \neq 0\}$ можна використати критерій Тьюки [1, с. 262, 263], що використовує статистику $S\ S_G$, вираз якої подано нижче у таблиці дисперсійного аналізу.

Зазначені дії виконуються у відповідності з таблицею ДА, що наводиться нижче.

Таблиця 2.7.2. **Таблиця дисперсійного аналізу при класифікації за двома ознаками для без- повторного експерименту з фіксованими ефектами**

Джерело дисперсії	Сума квадратів	Степені свободи	Середній квадрат
Фактор A	$SS_A = J \cdot \sum_{i=1}^{I} \hat{\alpha}_i^2$	$\nu_A = I - 1$	$Q_A = SS_A / \nu_A$
Фактор B	$SS_B = I \cdot \sum_{j=1}^{J} \hat{\beta}_j^2$	$\nu_B = J - 1$	$Q_B = SS_B / \nu_B$
Залишок	$SS_R = \sum_{i=1}^{I} \sum_{j=1}^{J} \hat{(\alpha\beta)}_{ij}^2$	$\nu_R = (I-1) \cdot (J-1)$	$Q_R = SS_R / \nu_R$
Повна	$SS_T = = \sum_{i=1}^{I} \sum_{j=1}^{J} (y_{ij} - \bar{y}_{..})^2$	$\nu_T = IJ - 1$	$Q_T = = SS_T / \nu_T$
Неадитивність G	$SS_G = \frac{(\sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i \beta_j y_{ij})^2}{\sum_i \hat{\alpha}_i^2 \sum_j \hat{\beta}_j^2}$	$\nu_G = 1$	$Q_G = SS_G / \nu_G = SS_G$

Неадитивність $A\,B$	$SS_{A\,B} = SS_R - SS_G$	$\nu_{A\,B} = I\,J - I - J$	$Q_{A\,B} = SS_{A\,B} / \nu_{A\,B}$

F- статистики для перевірки гіпотез:

$\boldsymbol{H_{A\,B}}$: $F_{A\,B} = Q_G / Q_{A\,B}$,

$\boldsymbol{H_A}$: $F_A = Q_A / Q_R$,

$\boldsymbol{H_B}$ **:** $F_B = Q_B / Q_R$.

Критичними точками є квантилі рівня $1 - \alpha$ розподілів Фішера, відповідно, з $(1, I\,J - I - J)$, $(I - 1, (I - 1)(J - 1))$, $(J - 1, (I - 1)(J - 1))$ степенями свободи.

2.7.4 Змішані двофакторні моделі і плани з рандомізованими блоками

Змішаною моделлю ДА називається модель, у якій одні фактори відповідають моделі I, а інші - моделі II. Для двох факторів формально можливі дві змішані моделі. Не втрачаючи загальності, у даному проекті будемо вважати, що фактор A з I рівнями відповідає моделі I, а фактор B з J рівнями — моделі II. Моделі ДА з вказаними планами експерименту мають назву *планів з рандомізованими блоками* (зміст такої назви роз'яснюється нижче).

Припустимо спочатку, що ми не повторюємо експериментів, так що $K = 1$. Тоді можемо записати змішаний двофакторний план у виді

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}\,,\ i = 1,\ldots, I,\ j = 1,\ldots, J. \qquad (5.37)$$

де μ - генеральне середнє, α_i є i -й диференціальний ефект фактору A, β_j — незалежні величини, розподілені по N (0, σ_b^2) (у випадку, коли за фактором B модель має тип 2, ефекти стовпців є центрованими нормально розподіленими випадковими величинами), ε_{ij} незалежні і розподілені по N (0, σ^2). Крім того, припускається, що величини β_j і $(\alpha\beta)_{ij}$ незалежні в сукупності і що між факторами A и B немає взаємодії.

Ця модель описує план експерименту, називаний планом без повторень (або безповторним планом) з рандомізованими блоками. Пояснимо зміст назви. Нехай дослідник хоче порівняти диференціальні ефекти α_i при I способах "обробки" (фактор A). Він випадково розподіляє їх по I експериментальних одиницях, однорідних по деякому параметру, що впливає на значення вимірюваної величини y. Ця множина з I одиниць називається блоком, а кожна одиниця — ділянкою. Весь експеримент повторюється J раз, тобто всі I способів випадковим образом розподіляються в кожнім з J блоків (фактор B). Саме цій схемі відповідають приведені вище модель і відповідна їй таблиця дисперсійного аналізу. Причому фактор A - це фактор, досліджуваний по моделі I, α_i - диференціальний ефект i-го способу обробки, фактор B визначає блоки і відповідає моделі II, σ_b^2 — дисперсія між блоками. А тому що кожен спосіб обробки застосовується тільки до однієї ділянки усередині блоку, то оцінити взаємодію "блок-обробка" неможливо. Тому і припускається, що між факторами A $и$ B взаємодія відсутня. Як і в кожній моделі дисперсійного аналізу, дана модель фіксує певний розподіл повної варіації ознаки y за впливами. А саме, мається на увазі розклад

$$SS_T = SS_A + SS_B + SS_R, \tag{5.38}$$

в якому складові мають наступні назви: SS_T — повна варіація, SS_A — міжгрупова варіація, SS_B — міжблочна варіація, SS_R — випадкова помилка. Вони визначаються наступними рівностями:

$$SS_T = \sum_{i=1}^{I}\sum_{j=1}^{J}(y_{ij} - \bar{y}_{..})^2 \,, SS_A = J \cdot \sum_{i=1}^{I}\hat{\alpha}_i^2 \,, SS_B = I \cdot \sum_{j=1}^{J}\hat{\beta}_j^2 \,, \quad (5.39)$$

$$SS_R = \sum_{i=1}^{I}\sum_{j=1}^{J}\hat{(\alpha\beta)}_{ij}^2 \,, \quad (5.40)$$

де наведені величини визначаються наступною групою рівностей

$$\hat{\mu} = \bar{y}_{..} \quad (5.41)$$

$$\hat{\alpha}_i = \bar{y}_{i.} - \bar{y}_{..} \,, \quad (5.42)$$

$$\hat{\beta}_j = \bar{y}_{.j} - \bar{y}_{..} \,, \quad (5.43)$$

$$\hat{(\alpha\beta)}_{ij} = y_{ij} - \bar{y}_{i.} - y_{.j} + \bar{y}_{..} \,, \quad (5.44)$$

$$\bar{y}_{i.} = \frac{1}{J}\sum_{j=1}^{J} y_{ij} \,, \quad (5.45)$$

$$\bar{y}_{.j} = \frac{1}{I}\sum_{i=1}^{I} y_{ij} \,, \quad (5.46)$$

$$\bar{y}_{..} = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij} \,. \quad (5.47)$$

Оцінки (5. 5) —(5.43) є незміщеними оцінками, відповідно, величин μ, α_i, β_j. Зауважимо, що величини $\bar{y}_{i\cdot}$, $\bar{y}_{\cdot j}$, $\bar{y}_{\cdot\cdot}$— це, відповідно, середнє по всім спостереженням у рядку i, середнє по всім спостереженням у стовпці j, середнє по всім спостереженням. Незміщеною оцінкою дисперсії (σ^2) випадкової складової ε_{ijk} є величина Q_R з наведеної нижче таблиці 1. 1.

Таблиця 2.7.3 **Таблиця дисперсійного аналізу (**ANOVA **)при класифікації за двома ознаками для безповторного експерименту з рандомізованими блоками**

Джерело дисперсії	Сума квадратів	Степені свободи	Середній квадрат
Фактор A	$SS_A = J \cdot \sum_{i=1}^{I} \hat{\alpha}_i^2$	$\nu_A = I - 1$	$Q_A = SS_A / \nu_A$
Фактор B	$SS_B = I \cdot \sum_{j=1}^{J} \hat{\beta}_j^2$	$\nu_B = J - 1$	$Q_B = SS_B / \nu_B$
Залишок	$SS_R = \sum_{i=1}^{I}\sum_{j=1}^{J} \widehat{(\alpha\beta)}_{ij}^2$	$\nu_R = (I-1) \cdot (J-1)$	$Q_R = SS_R / \nu_R$
Повна	$SS_T = = \sum_{i=1}^{I}\sum_{j=1}^{J} (y_{ij} - \bar{y}_{\cdot\cdot})^2$	$\nu_T = IJ - 1$	$Q_T = = SS_T / \nu_T$

Неадитивність G	$SS_G = \frac{(\sum_{i=1}^{I}\sum_{j=1}^{J}\alpha_i\beta_j y_{ij})^2}{\sum_i \hat{\alpha}_i^2 \sum_j \hat{\beta}_j^2}$	$\nu_G = 1$	$Q_G = SS_G / \nu_G = SS_G$
Неадитивність AB	$SS_{AB} = SS_R - SS_G$	$\nu_{AB} = IJ - I - J$	$Q_{AB} = SS_{AB} / \nu_{AB}$

Очікувані значення середніх квадратів *EMS* наведені в табл. 1.2.

Таблиця 2.7.4 **Математичні сподівання середніх квадратів для змішаної двофакторної моделі**

Джерело дисперсії	*EMS*
A: Модель *I* (між обробками)	$\sigma^2 + J(\sum \alpha_i^2)/(I-1)$
B: Модель II (між блоками)	$\sigma^2 + I\sigma_b^2$
R	σ^2

Незміщена оцінка дисперсії σ_b^2 задається формулою $s_{ab}^2 = (Q_B - Q_R)/I$; *F*-відношення для перевірки гіпотези H_0: всі α_i дорівнюють 0 — рівністю $F = Q_A / Q_R$, критичні точки при цьому — квантилі рівня $1 - \alpha$ розподілу Фішера $F(I-1, (I-1)(J-1))$. *F*- відношення для перевірки гіпотези H_0: $\sigma_b^2 = 0$ — рівністю $F = Q_B / Q_R$, критичні точки — квантилі рівня $1 - \alpha$ розподілу Фішера $F(J-1, (I-1)(J-1))$.

Наступний приклад пояснює термінологію і зміст зроблених вище означень.

Приклад 2.5.4.1 [1,2]. Дослідника цікавить оцінка і порівняння диференціальних ефектів *I* різновидів пшениці по величині врожайності. Але, оскільки різні поля можуть відрізнятися по родючості і тим самим впливати на врожайність пшениці, дослідник поділяє кожне поле на *J* блоків так, що кожен блок внутрішньо однорідний по родючості. Потім кожен блок поділяється на *I* ділянок, і кожна ділянка засівається своїм сортом пшениці. Якщо сорти розподіляються по ділянках усередині блоку випадково, то ми виявляємося в ситуації плану з рандомізованими блоками.

Повторюваному плану з рандомізованими блоками відповідає ситуація, коли кожен блок поділяється на *KI* ділянок, так що кожен сорт пшениці випадково приписується до *K* ділянок усередині блоку. У цьому випадку можна оцінити і взаємодію блок-обробка.

Модель, що описує повторюваний план з рандомізованими блоками, називається двофакторним змішаним повторюваним планом. Вона описується рівністю (5.1) Передбачається, що усі випадкові перемінні з цієї рівності в сукупності незалежні.

Приклад 2.5.4.2 [2] Оцінюються диференціальні ефекти дієти (4 варіанти) на предмет кількості азоту, що виділяється при диханні особою, якій дана дієта призначена. Було зафіксовано 4 дієти (d_1 — така, що не містить білків, d_2 — 23% білків, d_3 — 32% білків, d_4 — 67% білків) і 9 осіб, яким ці дієти були призначені.

В наступній таблиці 2.5.4 вказано спостережувані кількості азоту для всіх 9 × 4 = 36 досліджуваних одиниць, а також середні значення для кожної дієти Таблиця 2.7.5. **Набір даних**.

	d_1	d_2	d_3	d_4
	4,079	4,368	4,169	4,928
	4,859	5,668	5,709	5,608
	3,540	3,752	4,416	4,940
	5,047	5,848	5,666	5,291
	3,298	3,802	4,123	4,674
	4,679	4,844	5,059	5,038
	2,870	3,578	4,403	4,905
	4,648	5,393	4,496	5,208
	3,847	4,374	4,688	4,806
Середні	4,0963	4,6252	4,7477	5,0442

У цій ситуації будемо вважати кожну з 9 осіб «блоком», що одержує випадковим образом усі чотири дієти. Припустимо, що між двома "іспитами" проходить досить багато часу, так що перехідні ефекти дієти виключаються. Цей експеримент відповідає схемі неповторювального плану з рандомізованими блоками. Дані, приведені в таблиці 1.3, можна обробити

так, щоб одержати відповідну таблицю ANOVA (таблиця типу 1.1). Відзначимо, що, на відміну від двофакторного плану з фіксованими ефектами типу $A \times B$, досліджуються не 36 об'єктів, а 9. З цієї таблиці можна укласти, що по середній кількості видихуваного азоту досліджувані об'єкти значимо розрізняються, так само як і диференціальні ефекти, обумовлені дієтами. І взагалі "блокування" підвищує чутливість експерименту до фактору, що блокується. Відповідні чисельні результати наводяться у таблиці 2.5.4.

Таблиця2.7.6 **Таблиця ANOVA для аналізу ефектів чотирьох дієт методом рандомізованих блоків**

Джерело дисперсії	Сума квадратів	Степені свободи	Середній квадрат
Фактор A (дієта)	$SS_A = 4{,}4321$	$\nu_A = 3$	$Q_A = 1{,}4107$
Фактор B (особа)	$SS_B = 11{,}1236$	$\nu_B = 8$	$Q_B = 1{,}3902$
Залишок	$SS_R = 2{,}9353$	$\nu_R = 24$	$Q_R = 0{,}1223$
Повна	$SS_T = 18{,}2890$	$\nu_T = 35$	$Q_T = 0{,}5225$

F- відношення для перевірки значущості ефекту дієт і ефекту блоків дорівнюють відповідно: 11,53 і 11,37. Відповідні критичні значення в обох випадках дорівнюють 0,001 []. Бачимо, що по середній кількості видихання азоту значимо відрізняються як ефекти, що відповідають різним дієтам, так і ефекти, що відповідають досліджуваним індивідам («блокам»).

2.7.5. Комп'ютерні аспекти розв'язання задач ДА

Звернемося тепер до "комп'ютерної точки зору" і зосередимо увагу на програмах дисперсійного аналізу, що входять у стандартні пакети статистичних програм (ПСП). Багато пакетів програм містять єдину програму дисперсійного аналізу, що обчислює таблицю **ANOVA** для факторного плану. Нижче ми опишемо цей план, зупинимося на розходженнях між програмами, що допускають і не допускають повторення вимірів, і пояснимо, як можна аналізувати повторювані факторні плани, використовуючи програми другого типу.

Факторні програми можна використовувати і для аналізу інших типів планів, таких, як плани з рандомізованими блоками, повторюваними рандомізованими блоками і плани з групуванням. Ми опишемо також, як використовувати стандартну факторну програму при аналізі цих типів планів і ще двох — розщеплених і латинських квадратів.

2.7.5.1 Дисперсійний аналіз факторних планів

Нехай ми досліджуємо m факторів $A_1 ... , A_m$, і фактора A_i має усього $I_i \geq 2$ рівнів, $i = 1, ..., m$. Нехай кожна комбінація рівнів повторюється рівно N раз. Це значить, що кожен фактор сполучається з кожним, тобто заданий повний перехресний план. Якщо всі m факторів — фактори з фіксованими ефектами (моделі I), то говорять про (повний перехресний) факторний план з m факторами і з фіксованими ефектами. Якщо ж усі фактори — моделі II, то це — (повний перехресний) факторний план з m факторами з випадковими ефектами. Змішані моделі виникають, коли частина факторів — моделі I, а інші — моделі II. Поки що розглядатимемо тільки плани з фіксованими ефектами.

У моделі з фіксованими ефектами вважається, що виміри, котрі відповідають кожної комбінації рівнів $i_1\ i_2\ \dots\ i_m$ факторів $A_1\ A_2,\ ...,\ A_m$, представляються сумою середньої, що відповідає цій комбінації рівнів, і помилки виміру. Тоді

$$y_{i_1 i_2 \dots i_m\ n} = \mu_{i_1 i_2 \dots i_m} + \varepsilon_{i_1 i_2 \dots i_m\ n} \tag{5.48}$$

$$i_1 = 1,\dots, I_1, \dots, i_m = 1, ..., I_m, n = 1,\dots, N.$$

Для довільних індексів j, k, l = 1,2,…, m вважається, що середнє може бути представлене у вигляді суми

a_0) генерального середнього μ;

a_1) диференціального ефекту, що визначається фактором A_j; його позначення — $(\alpha_j)_{ij}$;

a_2) диференціального ефекту $(\alpha_j \alpha_k)_{i_j i_k}$, $j < k$, що визначається двофакторною взаємодією всіх пар різних факторів $A_j\ A_k$;

…….

a_m) диференціального ефекту $(\alpha_1\ \alpha_2\ \dots\ \alpha_m)_{i_1 i_2 \dots i_m}$, що визначається m-факторною взаємодією всіх m факторів $A_1 \dots , A_m$.

Як і раніш, будемо вважати, що величина помилки розподілена по N (0, σ^2). Для єдиності МНК- оцінок усіх параметрів потрібно накласти додаткові обмеження на самі параметри і їхні оцінки. Звичайно вимагають, щоб сума диференціальних ефектів для кожного фактора дорівнювала нулю, як і сума диференціальних ефектів для всіх k - факторних (k = 2, ..., m) взаємодій за кожним індексом при будь-яких фіксованих значеннях інших. Наприклад,

$$\sum_{i_j}(\alpha_j\alpha_k)_{i_j i_k} = 0 \text{ при всіх } i_k,\ \sum_{i_k}(\alpha_j\alpha_k)_{i_j i_k} = 0 \text{ при всіх } i_j \text{ і т.д.}$$

Використання факторних програм із ПСП особливо корисно при великих значеннях m, тому що навіть обчислення сум квадратів стає скрутним. Найчастіше ці програми називаються "дисперсійний аналіз факторного планування", "m- факторний дисперсійний аналіз", "дисперсійний аналіз m-факторних перехресних планів". Усі ці програми можна розбити на дві групи: такі, що припускають повторення експериментів (тобто $N \geq 1$) і такі, що не припускають ($N = 1$). Розглянемо тепер кожний з цих випадків.

1. **Програми, що допускають повторення**. У цьому випадку на виході типової факторної програми друкуються суми квадратів, ступені свободи, середні значення квадратів для залишкового компонента (чи помилки) і кожного джерела дисперсії A_1, A_2,…,A_m; A_1A_2, A_1A_3, …, $A_{m-1}A_m$; $A_1A_2\ldots A_m$, тобто зв'язаної з кожним фактором, взаємодією кожної пари різних факторів, кожної трійки попарно різних факторів і, нарешті, взаємодією усіх факторів. Для кожного джерела дисперсії (крім залишкової суми R дослідник може перевірити нульову гіпотезу H_0, що складається в тім, що усі відповідні диференціальні ефекти дорівнюють нулю. Наприклад, для джерела, що визначається взаємодією $A_1A_2A_3$ гіпотеза має вигляд H_0: $(\alpha_1\ \alpha_2\ \alpha_3)_{i_1 i_2 i_3} = 0$ при всіх i_1, i_2, i_3 . Всі очікувані значення середніх квадратів мають вигляд суми дисперсії помилки σ^2 і величин, що рівні 0, якщо тільки гіпотеза H_0 справедлива. А оскільки залишкова (нормована) сума квадратів Q_R дає незміщену оцінку σ^2, то для перевірки гіпотези H_0 ми розглянемо відношення середнього квадрата Q_H, що відповідає даному джерелу, до залишкового середнього Q_R. Нехай ν_H і ν_R позначають число ступенів свободи для цих

двох середніх квадратів. Тоді для перевірки виконання H_0 обчислюється F-відношення

$$F = Q_H / Q_R = S S_H / S S_R . \qquad (5.49)$$

Якщо гіпотеза H_0 справедлива, то ця статистика має F - розподіл зі ступенями свободи ν_H і ν_R. P- значення дорівнює площі під кривою щільності розподілу $F(\nu_H, \nu_R)$ праворуч від значення F (F- відношення). Гіпотеза H_0 відхиляється, якщо $P < \alpha$.Якщо число повторень в експерименті $N = 1$, то припускається, що всі диференціальні ефекти m- факторної взаємодії дорівнюють нулю. Тому у вихідній таблиці програми немає графи для джерела дисперсії $A_1 \ldots A_m$.

Програми, що передбачають неповторювальний експеримент.

Якщо в експерименті $N = 1$, то компонента, що відповідає m- факторній взаємодії, приймається за залишкову, так що

$$S S_R = S S_{A_1 A_2 \ldots A_m},\ \nu_R = \nu_{A_1 A_2 \ldots A_m},\ Q_R = Q_{A_1 A_2 \ldots A_m}.$$

Тому вихідні таблиці типових програм такого роду містять суми квадратів, число ступенів волі і середніх значень квадратів для усіх факторів і k-факторних взаємодій, $k = 2, \ldots, m$, але не містять графи для залишкового компонента. Для оцінки диференціальних ефектів можна скористатися співвідношенням (5.49).

З іншого боку, якщо в експерименті $N > 1$, то можна ввести новий фактор "повторення" A_{m+1} і скористатися програмою дисперсійного аналізу для $(m + 1)$- факторного плану, що одержано таким чином. На виході цієї програми ми одержимо дані для всіх джерел дисперсії і їхній k- факторних

взаємодій, $k = 2, ..., m, m + 1$. Потім необхідно об'єднати суми квадратів і число ступенів волі для фактора A_{m+1} і усіх взаємодій, що містять цей фактор. Ці об'єднані величини дають залишкову суму квадратів SS_R і число ступенів свободи для вихідного m- факторного плану з N повтореннями. Тому $Q_R = SS_R / \nu_R$, і ми знову можемо скористатися формулою (5.49) для оцінки ефектів.

Застосування факторних програм до інших моделей

Факторні програми можна використовувати для аналізу й інших видів планів, відмінних від повного m-факторного плану (див. статті Hartley у книзі Ralston, Wilf (I960)). Обов'язково потрібно тільки, щоб у всіх осередках було одне й те ж саме число спостережень N. Мистецтво планування експерименту полягає в тому, щоб сформулювати вихідний план як факторний, одержати таблицю дисперсійного аналізу для цього факторного плану, а потім виразити величини для вихідного плану, згрупувавши деякі суми квадратів і ступені волі факторного плану. Якщо це виконано, то середні значення квадратів знаходяться розподілом відповідної об'єднаної суми квадратів на її "об'єднане" число ступенів волі. По цим даним, як звичайно, проводиться перевірка гіпотез.

У цьому розділі ми розглянемо таку процедуру для двох раніше розглянутих планів — з рандомізованими блоками і з угрупованням. Крім того, ми опишемо ще два види планів — розщеплені плани і латинські квадрати. В міру необхідності ми будемо відзначати розходження між двома розглянутими категоріями факторних програм — що допускають і не допускають повторення.

1. *Плани з рандомізованими блоками*. Модель з рандомізованими блоками, що описується рівнянням (5.2), можна обробляти як факторний план із двома

факторами і $N = 1$ спостереженням для кожної пари рівнів. Відповідну таблицю дисперсійного аналізу видає будь-як факторна програма, а значення середніх квадратів обчислюються по формулах з табл. 1.2.

Модель повторюваного плану з рандомізованими блоками, описувану рівнянням (5.1), можна розглядати як повторюваний двухфакторный план з $N > 1$ і обробляти будь-якою факторною програмою, що допускає повторення спостережень. Якщо програма не допускає повторень, ми переформулюємо модель у такий спосіб: A_1 — фактор "спосіб обробки", A_2 — фактор "блок" і A_3 - фактор "повторення". Використовуючи факторну програму для трьохфакторного плану, ми можемо представити залишкову суму квадратів вихідної моделі у виді

$$S S_R = S S_{A_3} + S S_{A_1A_3} + S S_{A_2A_3} + S S_{A_1A_2A_3} , \qquad (5.50)$$

де величини, що фігурують у правій частині, беруться просто з таблиці дисперсійного аналізу для трьохфакторного плану. Аналогічна формула справедлива і для залишкового числа ступенів волі. Величини EMS задаються по формулах табл. 4.3.9 [1].

2. *Двофакторна модель з угрупованням*. У двофакторной моделі з угрупованням, що описується формулами (4.3.13) чи (4.3.14) [1], беруть участь два фактори A_1 і A_2, причому A_2 згрупований фактором A_1. У кожнім осередку виробляється N спостережень. При використанні факторної програми, що допускає повторення, можна розглядати наш план як факторний план із двомафакторами і N повтореннями. Сума квадратів для фактора A_2, підлеглого A_1 дорівнює

$$S S_{A_2(A_1)} = S S_{A_2} + S S_{A_1A_2} . \qquad (5.51)$$

Суми, що знаходяться у правій частині, містяться в таблиці дисперсійного аналізу для факторного плану. Аналогічна формула є вірною і для числа ступенів волі $\nu_{A_2(A_1)}$. Величини SS_A, і SS_R, так само як і відповідні їм числа ступенів волі, беруться прямо з цієї таблиці.

При використанні програми, що не допускає повторень, уведемо фактор "повторень" A_3 і розглянемо отриману модель як трьохфакторный план. Залишкова сума квадратів для моделі з угрупованням виражається через величини, видавані програмою по формулі

$$SS_R = SS_{A_2} + SS_{A_1A_2} + SS_{A_1A_2A_3}. \qquad (5.52)$$

Число ступенів волі знаходиться аналогічно. Величина

$SS_{A_2(A_1)}$ задається рівністю (5.51), SS_{A_1} і ν_{A_1} видаються безпосередньо програмою. Значення *EMS* відповідають табл. 4.3.12 [2].

3. *План з розщепленими блоками*. У цій ситуації ми маємо I_1 видів обробки (фактор A_1), I_2 підвидів обробки (фактор A_2) і I_3 блоками (фактор A_3) ([2]). Кожен блок поділяється на I_1 однорідних ділянок, а кожна ділянка — на I_2 підділянок. Усередині кожного блоку рівні фактора A_1 випадково розподіляються по ділянках, а усередині ділянки рівні фактора A_2 випадково розподіляються по підділянках. Наприклад, один блок у випадку $I_1 = 3$, $I_2 = 2$ може бути влаштований таким чином, як вказано у наступній таблиці (див. [2]) , в якій індекс i_j позначає рівні фактору j, $j = 1,2$.

Ділянка I	Ділянка II	Ділянка III	
$i_1 = 2$	$i_1 = 3$	$i_1 = 1$	
$i_2 = 3$	$i_2 = 2$	$i_2 = 1$	Підділянка 1

$i_1 = 2$	$i_1 = 3$	$i_1 = 1$	Підділянка 2
$i_2 = 1$	$i_2 = 1$	$i_2 = 2$	
	Блок		

Мета такого плану складається в зменшенні числа комбінацій способів обробки усередині одного блоку. Моделлю плану служить співвідношення

$$y_{i_1 i_2 i_3} = \mu + (\alpha_1)_{i_1} + (\alpha_2)_{i_2} + (\alpha_3)_{i_3} + (\alpha_1\alpha_2)_{i_1 i_2} + (\alpha_2\alpha_3)_{i_2 i_3} + \varepsilon^{(1)}_{i_1(i_3)} + \varepsilon_{i_1 i_2 i_3} . \quad (5.53)$$

Тут α_1 - фіксовані ефекти, обумовлені видами обробки, α_2 - фіксовані ефекти, обумовлені підвидами, α_3 - випадкові ефекти блоків, $\alpha_1\alpha_2$ - взаємодія виду і підвиду обробки, $\alpha_2\alpha_3$ - взаємодія підвиду з блоком. Член $\varepsilon^{(1)}$ - випадкова помилка ділянок усередині блоку, а ε - випадкова помилка підділянки усередині ділянки. Припускається, що $\varepsilon^{(1)}$ розподілено за законом $N(0, \sigma_1^2)$, а ε - за $N(0, \sigma^2)$. Додаткові обмеження мають вигляд

$$\sum_{i_1}(\alpha_1)_{i_1} = \sum_{i_2}(\alpha_2)_{i_2} = 0, \ \sum_{i_1}(\alpha_1\alpha_2)_{i_1 i_2} = \sum_{i_2}(\alpha_1\alpha_2)_{i_1 i_2} = 0.$$

Таблиця дисперсійного аналізу для такого плану наводиться знайти в [2] табл. 4.4.1, а критерії перевірки гіпотез наведені в тому ж джерелі, табл. 4.4.2. Якщо гіпотеза H_0 не відкидається, то прихильник об'єднання може перейти до об'єднаних оцінок сум $SS_R^{(1)}$ і $S\ S_R$, щоб одержати нову залишкову суму квадратів. Ця сума використовується при перевірці гіпотез H_0: $(\alpha_1)_{i_1} = 0$ і H_0: $\sigma^2_{a_2} = 0$. Щоб одержати табл. типу 4.4.1 [2], використовуючи факторну

програму, ми розглянемо розщеплений план як план з трьома факторами при $N = 1$ спостереженнях для кожної трійки рівнів. Співвідношення між двома наборами джерел дисперсії (факторної і розщепленої моделі) можна знайти в [2].

Література по розділу 2

1. Anderson T,W. An Introduction to Multivariate Statistical Analysis, New York — J. Willey and Suns Incorp., 1965
2. Afifi A.A, Azen S.P. Statistical Analysis. A Computer Oriented Approach. — Second Edition, Academic Press, New York – San Francisko – London, 1979. – 488 p.
3. Draper N.R., Smith H. Applied Regression Analysis – Third Addition, J. Willey and Suns Incorp, New York – Chichester—Singapure—Toronto, 2005 – 911 p.
4. George A. F. Seber, Alan J. Lee Linear Regression Analysis 2nd Edition - John Wiley & Sons, Inc., Hoboken, New Jersey, 2013.
5. Levine D.M., Stephan D., Krehbiel T.C., Berenson M. Statistics for Managers, Prentice Hall, Upper Saddle River, New Jersey 07458, 2005 — 1810 p.
6. D. Snider, E. B. Saff Fundamentals of Matrix Analysis with Applicationsю John Wiley & Sons, Inc., Hoboken, 2015
7. Tukey J.W. Exploratory Data Analysis Addison Wesley Publishing Company Reading, Massachusets, 2018 — 693 p.
8. Mardia R., Zemroch P. Tables of the F- and Related Distributions With Algorithms, Academic Press, 1978, — 250 p.
9. Краснитський С.М.,Резанова В.Г. Регресійний аналіз. — К.: КНУТД, 2007 — 47 с.
10. Краснитський С.М., Щербань В.Ю. та ін. Векторні випадкові величини і випадкові процеси. — К.: Конус-Ю, 2008. — 191 с.
11. Краснитський С.М., Щербань В.Ю., Резанова В.Г. Наукова інформатика. — К.: КНУТД, 2014.
12. Краснитський С.М., Щербань В.Ю. Прогнозування технологічних процесів. — К.: КНУТД, 2016.
13. С.М. Краснитський, В. Ю. Щербань, В. Г. Резанова. Ймовірнісні процеси та математична статистика в легкій промисловості. Основні поняття математичної статистики: методичні вказівки до лабораторних робіт для студентів денної та заочної форм навчання напряму «Комп'ютерні науки» – К. : КНУТД, 2016. – 95 с.

14. Щербань В.Ю., Краснитський С.М., Резанова В.Г. Математичні моделі в САПР. — К.: КНУТД, 2011 — 219 с.
15. Оленко А.Я. Комп'ютерна статистика. К.: ВПЦ «Київський університет», 2007 — 174 с.
16. Масик С.В., Кукуш О.Г.. Шкляр С.В. Чепурний М.І., Ліхтарьов І.А. Моделі регресії з похибками вимірювання та їх застосування до оцінювання радіаційних ризиків. — К.:, ДІА, 2015. — 287 с.
17. Douglas C. Montgomery Design and Analysis of Experiments": Wiley, 2017 — 736 p

3. WEB - ТЕХНОЛОГІЇ

3.1. Клієнт-серверні технології.

3.1.1. Поняття та складові веб- технологій

Веб-технологія *– це сукупність методів та програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі (локальному або глобальному, наприклад, мережі Інтернет).*

Поняття веб-технології пов'язане з використанням веб-простору – WWW (англ. World Wide Web) — глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP(рис.1.1).

Веб-технологія — це створення та використання механізмів, які дозволяють різним комп'ютерам і пристроям обмінюватися ресурсами та розповсюджувати їх. Веб-технології є будівельними блоками інфраструктури будь-якої ефективної комп'ютерної мережі

Рисунок 3.1.1.- Веб-технологія

За допомогою засобів веб-технології можна опрацьовувати різні види інформації – текстову, графічну, звукові та іншу, що знаходиться в мережі. Якби це питання стосувалось інформації на комп'ютері користувача (що не має підключення до мережі), то можна використовувати текстові, графічні, звукові редактори. А для здійснення цих операцій в мережі достатньо використовувати веб-оглядач (браузер) і набір відповідних веб-технологій, які функціонують в мережі на різних серверах.

Наприклад, для того щоб ввести текст електронного листа не потрібно використовувати текстовий редактор, що встановлено на комп'ютері, адже в інтерфейсі електронної пошти є вбудований редактор, який дає змогу здійснювати набір тексту в браузері. Така технологія називається *веб-технологією*. Наприклад, всім відомі соціальні мережі використовують різні веб-технології, які дають змогу завантажувати і переглядати фотографії, музичні файли, відео тощо.

Система Google пропонує сервіс Google-docs, який має текстовий веб-редактор для створення документів у форматі *.doc,* веб-редактор презентацій та електронних таблиць. Виходячи з цього, людині не потрібно використовувати програмне забезпечення свого комп'ютера, достатньо відкрити веб-оглядач, ввести адресу системи Google та створити потрібний документ.

Будь-яку веб-технологію можна реалізувати засобами мережі – чи глобальної, чи локальної, в межах однієї аудиторії чи цілого корпусу.

Основні принципи функціонування мережі Інтернет.

Всесвітню мережу утворюють мільйони **веб-серверів** мережі Інтернет, розташованих по всьому світу.

Веб-сервер є програмою, що запускається на підключеному до мережі комп'ютері і використовує *протокол HTTP* для передачі даних.

У простому вигляді така програма отримує по мережі *HTTP-запит* на певний ресурс, знаходить відповідний файл на локальному жорсткому

диску і відправляє його по мережі до того комп'ютера, який робив запит. Складніші веб-сервери здатні динамічно формувати ресурси у відповідь на HTTP-запит. Приклад такого веб-сервера можна назвати будь-яку пошукову систему, яка формує список веб-ресурсів на запит користувача.
Для *ідентифікації ресурсів* (часто файлів або їх частин) у Всесвітній павутині використовуються одноманітні (форменні) ідентифікатори ресурсів *URI* (англ. Uniform Resource Identifier) – це послідовність символів, що ідентифікує абстрактний або фізичний ресурс.

Для визначення місцезнаходження ресурсів в мережі використовуються одноманітні локатори ресурсів URL (англ. Uniform Resource Locator). Такі URL-локатори поєднують в собі технологію ідентифікації URI і систему доменних імен DNS (англ. Domain Name System).

Доменне ім'я (або безпосередньо IP-адреса в числовому записі) входить до складу URL для позначення комп'ютера (точніше — одного з його мережевих інтерфейсів), який виконує код потрібного веб-сервера.

Доменне ім'я - це унікальне алфавітно-цифрове ім'я, що ідентифікує конкретний вузол мережі Інтернет. Якщо говорити про хостінг, то доменне ім'я - це унікальна адреса, за допомогою якої будь-який користувач мережі Інтернет може знайти ресурс в Інтернеті.
Доменне ім'я складається з назви сайту та доменної зони**.** В імені можуть використовуватись латинський алфавіт, цифри, дефіс. При створенні ім'я не може використовуватись нижнє підкреслення (_), апостроф ('), службові слова (www, ping...). В багатьох зонах заборонені 2-буквені імена. Але в зонах, де вони дозволені, весь набір 2-буквених імен давно вичерпно.

Зони бувають *міжнародні* та *національні* (вказують на певну країну). Національні, в свою чергу, поділяються за регіональною приналежності і за сферою діяльності.

Всі міжнародні доменні імена відносяться до імен I рівня (зона складається з одного слова, наприклад, google.com, wikipedia.org.

Приклади *доменів I рівня*: .com, .net, .org, .biz, .info, .gov, .edu, .mil

Національні доменні імена можуть бути як I рівня, так і II чи III. Прикладом національного доменного імені I рівня можуть бути імена: kiev.ua, , meta.ua.

Для перегляду інформації, отриманої від веб-сервера, на клієнтському комп'ютері застосовується спеціальна програма — веб-браузер. Основна функція веб-сервера-браузера — відображення *гіпертексту*.

Всесвітня павутина нерозривно пов'язана з поняттями гіпертексту і гіперпосилання. Більша частина інформації у веб-просторі являє собою саме гіпертекст.

Термін гіпертекст був введений Тедом Нельсоном в 1965 році для позначення «тексту, що розгалужується або виконується, за запитом». Зазвичай *гіпертекст* - це набір текстів, що містять вузли переходу від одного тексту до іншого. Це дає змогу обирати об'єкт або послідовність читання. Загальновідомим і яскраво вираженим прикладом гіпертексту є веб-сторінки — документи створені мовою HTML (гіпертекстовій мові розмітки), розміщені в мережі.

Гіперпосилання – це елемент веб-документу (текст, картинка), що містить посилання на інший веб-ресурс мережі.

Для полегшення створення, зберігання і відображення гіпертексту у Всесвітній мережі традиційно використовується мова HTML (англ. Hypertext Markup Language), мова розмітки гіпертексту. Робота з розмітки гіпертексту називається верстkою, майстра розмітки називають веб-майстром. Після HTML-розмітки гіпертекст, вміщується у файл, такий HTML-файл є найпоширенішим ресурсом Всесвітньої мережі. Після того, як HTML-файл стає доступним на веб-сервері, його починають називати *«веб-сторінкою»*. Набір веб-сторінок утворює веб-сайт. До тексту веб-

сторінок додаються гіперпосилання. Гіперпосилання допомагають користувачам Всесвітньої мережі легко переміщатися між ресурсами (файлами) незалежно від того, знаходяться ресурси на локальному комп'ютері або на віддаленому сервері. Гіперпосилання веб-простору засновані на технології URL.

В цілому можна зробити висновок, що всесвітня павутина (WWW) будується на основі великої кількості різноманітних технологій, які забезпечують ту чи іншу функцію в мережі Інетрент. Наприклад, для створення динамічних сторінок використовуються мову PHP, для поліпшення візуального сприйняття веб-протору – технологія CSS, яка дозволяє задавати єдині стилі оформлення для багатьох веб-сторінок одразу, тощо.

Веб-сайт, або просто сайт (англ. website, від web — мережа і site — «місце») — це сукупність веб-сторінок, доступних в мережі через протоколи HTTP/HTTPS; сукупність всіх загальнодоступних веб-сайтів і є всесвітня павутина. Сторінки веб-сайту об'єднані загальною кореневою адресою, а також темою, логічною структурою, оформленням і/або авторством.

Раніше поняття сайту плутали з фізичним вузлом мережі — *хостом, сервером* (вузлом). Але із розвитком мережі Інтернету і технологічним поліпшенням серверів на одному комп'ютері стало можливе розміщення безлічі сайтів і доменів.

Сторінки веб-сайтів — це файли з текстом, розміченим на мові HTML або XHTML. Ці файли завантажуються відвідувачем мережі на його комп'ютер, обробляються програмою-браузером і виводяться на засіб відображення користувача (монітор, екран КПК, принтер, тощо). Мова HTML/XHTML дозволяє форматувати текст, розрізняти в ньому функціональні елементи, створювати гіпертекстові посилання (гіперпосилання) і вставляти в сторінку зображення, звук та інші

мультимедійні елементи. Відображення сторінки можна змінити додаванням до неї таблиці стилів на мові CSS або сценаріїв на мові JavaScript. Мова JavaScript дає змогу створювати на сторінці динамічні об'єкти, що прикрашають та роблять більш привабливою сторінкою.[трофимч]

Сторінки сайтів можуть бути простими статичними наборами файлів або створюватися спеціальною комп'ютерною програмою на сервері — так званим «движком» сайту. Така програма може бути або зроблена на замовлення для окремого сайту, або готовим продуктом, розрахованим на певний різновид сайтів. Деякі з програм можуть забезпечити власнику сайту можливість гнучкого налаштування структуризації і виведення інформації на веб-сайті; такі програми називаються системами управління змістом.

Виготовлення сайтів як працюючих цілісних інформаційних ресурсів – це складний процес, що потребує поєднання різних професійних навичок. Загальний термін на позначення сайтобудування — *«веб-розробка».*

Веб-сервіси – це технологія, яка дозволяє додаткам обмінюватися даними незалежно від платформи і мови програмування. Сервіс – послуга, що забезпечує певну потребу людини.

Веб-сервіс обов'язково має програмний інтерфейс, який отримує через мережу команди і дані в заздалегідь обумовленому форматі, виконує якісь операції і відправляє через мережу відповідь. Дані, що передаються мережею мають один із загальноприйнятих форматів зазвичай це якийсь різновид *XML.* Як протокол практично завжди використовується TCP/IP, а якщо точніше, то HTTP або HTTPS.

Група веб-сервісів, які взаємодіють в описаній вище манері складають *веб-додаток.* Відповідна архітектура додатка називається

орієнтованою на сервіси. Сьогодні в мережі інтернет функціонує велика кількість веб-сервісів мережі Інтернет, які засновані на технології ВЕБ 2.0.

Технології Веб 2.0 називають *соціальними сервісами мережі* Інтернет, оскільки їх використання, зазвичай, здійснюється спільно в межах відповідної групи користувачів. Групи користувачів можуть утворювати цілі мережні співтовариства, які об'єднують свої зусилля для досягнення відповідної мети. Прикладом такої групи може бути створення мережного співтовариства студентів – майбутніх ІТ для спільного використання ресурсів

Технології Веб 2.0 дають змогу не лише переглядати веб-ресурси мережі Інтернет, а й завантажувати власні, здійснювати обмін цими ресурсами з іншими користувачами, діяти спільно з метою накопичення корисних веб-ресурсів, брати участь в обговореннях та ін.

Якщо розглядати Веб 2.0 з технологічного боку, то доцільно згадати мови програмування JavaScript та мову розмітки тексту XML, які в поєднанні утворюють новий підхід до побудови *веб-додатків – AJAX* (від англ. Asynchronous JavaScript and XML – асинхронний JavaScript та XML). Цей підхід полягає в тому, що обмін даними між веб-оглядачем і веб-сервером відбувається в «фоновому» режимі, в результаті чого під час оновлення даних веб-сторінка не перезавантажується повністю.

3.1.2. Архітектура клієнт-серверні технології

Веб-додаток – це програмне забезпечення, яке можна відкрити за допомогою будь-якого браузера. Ключова відмінність веб-додатків від веб-сайтів полягає у тому, що вони відображають не статичні дані в Інтернеті такі, як тексти та зображення, а динамічні програми.

Клієнт-серверна конфігурація - це концепція інформаційної мережі, в якій, з одного боку, клієнти надсилають запити, з іншого боку, сервер відповідає на надіслані запити.

Веб-додаток можна умовно розділити на:

- клієнтську частину,
- серверну частину
- інформаційне наповнення (контент)

Отримання необхідних масивів даних виконується в інформаційних системах, в яких міститься клієнт-серверну архітектуру. Це означає, що програма або сервіс складається з двох програмних частин – клієнтської та серверної.

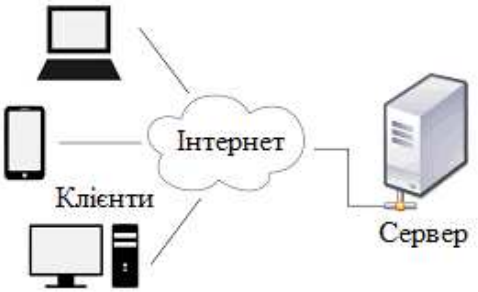


Рисунок 3.1.2. Клієнт-серверна архітектура

Серверна частина веб-додатку (*Back-end*) – це основна частина програми, яка виконується на основі інтернет-запитів користувачів, надісланих через веб-браузери. Серверний компонент складається з двох частин: 1) логіки додатка та 2) бази даних. Логіка додатку – це головний центр керування веб-додатком, а бази даних – це місце, де зберігається інформація.

Серверний програмний код знаходиться на веб-сервері та відповідає на *HTTP*-запити клієнтів, створюючи сторінку. Він, також, відповідає за зберігання різного роду даних, зокрема профілі клієнтів. Для написання коду використовують *JavaScript*, *PHP*, *Python*, тощо.

Клієнтська частина веб-додатку (*Front-end*) – це інтерактивна частина програми, що виконується в веб-браузері на різних пристроях - комп'ютері, смартфоні або планшеті користувача. Клієнтський компонент активує інтерфейс користувача веб-додатку і завантажується на пристрої у

вигляді динамічних веб-сторінок. Веб-додатки запускаються на будь-яких пристроях та операційних системах, де є інтернет-браузери. Для написання клієнтського коду використовують комбінацію *JavaScript* (головна мова програмування), *CSS* (надає веб-сторінці певний вигляд), *HTML* (відповідає за структурування змісту сторінки). Вони вбудовані у більшість браузерів, тому немає потреби окремо налаштовувати пристрій. Клієнтський код взаємодіє з сервером лише через *HTTP*, й не може безпосередньо отримувати інформацію з сервера.

База даних веб-додатку – це масиви даних, в яких міститься необхідна клієнтам інформація. Серверна частина веб-додатку за запитом користувача використовує ці дані для обробки та надає бажаний результат кінцевому клієнту.

Система, яка заснована на взаємодії клієнт-сервер, містить в собі три основні компоненти: компонент керування ресурсами та їх зберігання, представлення даних та прикладний компонент.

Архітектура «клієнт-сервер» ділить додаток на дві частини, «клієнт» та «сервер».

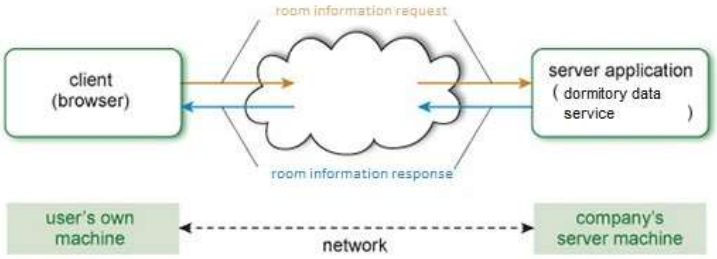


Рисунок 3.1.3 – Схема клієнт-серверної архітектури

Такий додаток реалізований в комп'ютерній мережі, яка підключає клієнта до сервера. Серверна частина цієї архітектури забезпечує центральну функціональність: тобто будь-яка кількість клієнтів може

підключитися до сервера і вимагати, щоб він виконував завдання. Сервер приймає ці запити, виконує необхідне завдання та, відповідно, повертає будь-які результати клієнту.

В якості прикладу можна розглянути інтернет-книгарню. Додаток дозволяє користувачеві шукати та переглядати деталі великого асортименту книг, а потім замовляти книги. Прикладне програмне забезпечення забезпечує інтерфейс та засоби вибору чи пошуку деталей книги, а також відображає інформацію про книгу та дозволяє генерувати замовлення книги.

Додаток може мати форму єдиного «шматка» програмного забезпечення, завантаженого з Інтернету. Однак, якщо програмне забезпечення є одним монолітним елементом, то кожен раз, коли що-небудь змінюється або оновлюється, вся програма повинна бути перерозподілена знову. Очевидно, що це не буде працювати в цьому прикладі, оскільки каталог книг буде регулярно змінюватися. Поліпшення може полягати в розділенні програми на дві частини. Одна частина, клієнт, може надати інтерфейс для користувачів і поширюватися серед них. Іншу частину можна зберігати та запускати на власному сервері компанії (рис.1.3).

Клієнтська програма може відображати інформацію і використовуватись для передачі інформації на сервер для пошуку, наприклад, заголовка книги. Цю клієнтську програму або програмне забезпечення досить часто називають «презентаційним» рівнем .

У цій моделі багато клієнтів можуть підключатися до серверної програми та запитувати інформацію про книги. Сервер повинен обробити ці запити та надіслати відповідь клієнту, який ініціював запит, а не будь-якому іншому клієнту. Поки мережа працює добре, і сервер може не відставати у відповіді на всі запити, які він отримує, такий «розділений» додаток забезпечить майже такий же рівень обслуговування, як і монолітна

версія. Цю просту архітектуру клієнт-сервер також зазвичай називають «дворівневою архітектурою».

Дворівнева архітектура складається із серверу, який відповідає за отримання та відправлення відповідей клієнту, не використовуючи сторонні ресурси, та з клієнта, який представляє інтерфейс користувача.

Принцип роботи трирівневої архітектури полягає у розподіленні операцій між серверами, що знижує навантаження на них.

Трирівневу систему можна розшири до багаторівневої. Для цього потрібно встановити додаткові сервери. Така архітектура дозволяє збільшити ефективність роботи інформаційних систем та оптимізувати розподілення її програмно-апаратних ресурсів.

Існує концепція побудови клієнт-серверної системи, коли існуючих клієнтів можна умовно можно поділити на два підтипи: товстий та тонкий. Також є системні архітектури, в яких характеристики цих підтипів поєднуються і перетинаються їх називають «гібридним» *клієнтом.*

1.Товстий клієнт – це клієнт, який забезпечує повну функціональність та незалежність програми від центрального сервера**.** У цьому випадку сам сервер виконує роль звичайного сховища даних, а все навантаження, пов’язане з обробкою та поданням інформації лягає на пристрій клієнта.

2. Тонкий клієнт. Тонкий клієнт не виконує жодних завдань, пов’язаних з обробкою даних. Натомість всі обчислювальні потужності переносяться на віддалений сервер, з яким він взаємодіє за допомогою термінального доступу**.** У такому випадку єдине завдання клієнта полягає в запуску мережного програмного забезпечення.Через термінальне з’єднання він зв’язується з основним потужним комп’ютером, що виконує роль

сервера, на якому зберігаються всі дані та запущені програми, з якими взаємодіє користувач.

Таку архітектуру використовують багато популярних сервісів, включаючи Google Drive, WP та браузерні онлайн-ігри.

Серверне програмне забезпечення може включати один або кілька сховищ даних (наприклад, у формі системи баз даних). На прикладі книжкового магазину, один сховище даних може включати зображення, вартість та відгуки про книгу; інша може використовуватися для зберігання більш динамічної інформації, наприклад, поточного рівня запасів або часу замовлення для кожної книги. Розбиття даних таким чином, наприклад, дозволило б періодично створювати резервні копії більш статичних даних з більш частими резервними копіями динамічних даних. Отже, архітектура, яка використовується в Інтернеті, може виглядати в таких контурах, як показано на рисунку 1.4.

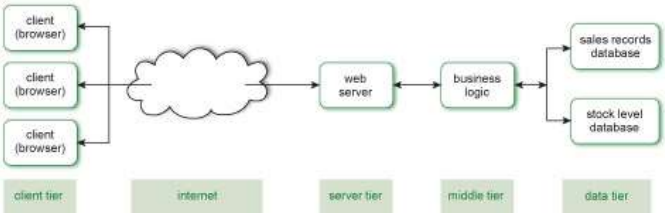


Рисунок 3.1.4 Схема багаторівневої архітектури

Тут ми додали ще два рівні на стороні сервера: рівень даних та інший рівень, який обробляє взаємодію з рівнем даних, наприклад отримання запитуваної інформації або перевірка даних, які зберігаються у сховищах даних. Цей рівень між рівнем даних та рівнем сервера іноді називають «середнім рівнем» або проміжним програмним забезпеченням.

Програма, яка використовує проміжне програмне забезпечення для обробки запитів даних між користувачем та базою даних, використовує *багаторівневу архітектуру*.

Як клієнтську, так і серверну частини можна додатково розділити, якщо це підходить для програми. Наприклад, клієнт може нести відповідальність як за певну обробку отриманих даних, так і за подання інформації. У прикладі книгарні нам може знадобитися розрахувати загальну вартість замовлення та відобразити зразок вмісту книг. Ці дві функції можна розділити на два рівні на стороні клієнта

Досить часто «багаторівнева архітектура» відноситься до того, що конкретніше слід називати трирівневою архітектурою (клієнт, сервер та рівні даних). Однак можна використовувати більше рівнів і тому термін «архітектура N-рівня» зазвичай використовується для позначення будь-якої архітектури, яка має більше двох рівнів.

Кожен рівень можна змінити легше, оскільки він менш залежить від точних деталей інших рівнів, з якими він взаємодіє. Це знову залежить від того, наскільки обережно ми застосовуємо підхід, який гарантує, що рівні або компоненти дійсно вільно зв’язані. Просто розбиття програми на шматки не гарантує цього; нам також потрібно прийняти відповідні стандарти та визначити точні та обмежені взаємодії між рівнями. Архітектура N-рівня майже стала всеосяжним підходом, і, безумовно, є дуже поширеною. Зараз доступні набагато більше клієнтів, крім веб-браузерів, які реально можна обмінюватись без зайвих зусиль, включаючи бази даних та веб-сервери.

3.1.3 Сервісно-орієнтовні архітектури. Види протрколів

SOA - це архітектурне рішення для інтеграції різноманітних систем шляхом надання архітектурного стилю, що сприяє вільному зчепленню та повторному використанню. Програмні компоненти надають сервіси іншим компонентам через протокол зв'язку, як правило, через мережу Інтернет.

Сторона, що пропонує сервіс, відома як постачальник послуг (сервер), а сторона, що викликає послугу, споживача послуги (клієнт).

Сервіс - це деяка функціональність, як правило, бізнес-процес, який упаковується як багаторазовий програмний компонент.

REST(Representational state transfer). Для побудови взаємодії між сервером та клієнтом часто обирають REST архітектуру. REST - це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови веб-служб. Термін REST був введений у 2000 році Роєм Філдінгом, одним з авторів HTTP-протоколу. Системи, що підтримують REST, називаються RESTful-системами.

У загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання додаткових внутрішніх прошарків. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожна URL в свою чергу має строго заданий формат.

Ідентифікатор ресурсу URI (Uniform Resource Identifier) представляє собою коротку послідовність символів, що ідентифікує абстрактний або фізичний ресурс. URI не вказує на те, як отримати ресурс, а тільки ідентифікує його. Це дає можливість описувати за допомогою RDF (Resource Description Framework) ресурси, які не можуть бути отримані через Інтернет (імена, назви, тощо). Найвідоміші приклади URI - це URL і URN.

- URL (Uniform Resource Locator) - це URI, який, крім ідентифікації ресурсу, надає ще й інформацію про місцезнаходження цього ресурсу.
- URN (Uniform Resource Name) - це URI, який ідентифікує ресурс в певному просторі назв, але, на відміну від URL, URN не вказує на місцезнаходження цього ресурсу.

URL має наступну структуру:

<Схема>: // <логін>: <пароль> @ <хост>: <порт> / <URL - шлях>

де:

- *схема* - схема звернення до ресурсу (зазвичай мережевий протокол);
- *логін* - ім'я користувача, що використовується для доступу до ресурсу;
- *пароль* - пароль, асоційований з вказаним ім'ям користувача;
- *хост* - повністю прописане доменне ім'я хоста в системі DNS або IP-адреса хоста;
- *порт* - порт хоста для підключення;
- *URL-шлях* - уточнююча інформація про місце знаходження ресурсу.

Загальноприйняті схеми (протоколи) URL включають протоколи: *ftp, http, https, telnet* , а також:

- *gopher* - протокол Gopher;
- *mailto* - адреса електронної пошти;
- *news* - новини Usenet;
- *nntp* - новини Usenet через протокол NNTP;
- *irc* - протокол IRC;
- *prospero* - служба каталогів Prospero Directory Service;
- *wais* - база даних системи WAIS;
- *xmpp* - протокол XMPP (частина Jabber);
- *file* - ім'я локального файлу;
- *data* - безпосередні дані (Data: URL).

Клієнт-серверна архітектура визначає принципи спілкування між машинами за правилами, які визначено у мережевому протоколі.

Існують наступні мережеві протоколи:

- TCP – протокол, який використовують для встановлення надійного зв’язку між двома вузлами, передачі даних та отримання підтвердження, що дані отримано.

- TCP/IP – це стек протоколів передачі даних, визначення всієї мережі, яка працює на основі двох протоколів – TCP і IP.
- IP – інтернет протокол, який відповідає за те, щоб повідомлення були доставлені за правильною адресою, розбиваючи їх на пакети.
- UDP – протокол, який керує передачею даних, але не перевіряє інформацію при доставці.
- POP3 – стандартний протокол, який відповідає за доставку пошти.
- SMTP – протокол, який перевизначає правила для передачі пошти. Відповідає за сповіщення про помилки, повернення або підтвердження доставки.
- ICMP – протокол, який не використовують для передачі даних, але який відповідає за обмін інформацією.
- HTTP – прокол, на основі якого працюють всі сайти, його функція – передача гіпертексту.
- FTP – протокол передачі файлів на комп'ютер із файлового серверу.
- SSH – протокол для забезпечення віддаленого курування системою через захищений канал.
- SMTP – протокол, який перевизначає правила для передачі пошти. Відповідає за сповіщення про помилки, повернення або підтвердження доставки.
- MAC – протокол, за допомогою якого відбувається ідентифікація пристроїв в мережі.

3.1.4 Протокол прикладного рівня .

HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного рівня для передачі гіпертексту.

Центральним об'єктом в HTTP є ресурс, на який вказує URI в запиті клієнта. Зазвичай такими ресурсами є файли які зберігаються на сервері.

Особливістю протоколу HTTP є можливість вказати в запиті і відповіді спосіб представлення одного і того ж ресурсу за різними параметрами: формату, кодуванні, мови і т. Д. Саме завдяки можливості вказівки способу кодування повідомлення клієнт і сервер можуть обмінюватися двійковими даними, хоча спочатку даний протокол призначений для передачі символьної інформації. На перший погляд це може здатися зайвою тратою ресурсів. Дійсно, дані в символьному вигляді займають більше пам'яті, повідомлення створюють додаткове навантаження на канали зв'язку, однак подібний формат має багато переваг. Повідомлення, що передаються по мережі, зручні для читання, і, проаналізувавши отримані дані, системний адміністратор може легко знайти помилку і усунути її. При необхідності роль одного з взаємодіючих додатків може виконувати людина, вручну вводячи повідомлення в необхідному форматі.

На відміну від багатьох інших протоколів, HTTP є протоколом без пам'яті. Це означає, що протокол не зберігає інформацію про попередні запитах клієнтів і відповідях сервера. Компоненти, використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаної з останніми запитами і відповідями. Наприклад, клієнтський веб-додаток, що посилає запити, може відстежувати затримки відповідей, а веб-сервер може зберігати IP-адреси і заголовки запитів останніх клієнтів

Все програмне забезпечення для роботи з протоколом HTTP поділяється на три основні категорії:

- Сервери - постачальники послуг зберігання і обробки інформації (обробка запитів).
- Клієнти - кінцеві споживачі послуг сервера (відправка запитів).
- Проксі-сервери для підтримки роботи транспортних служб.

Основними клієнтами є браузери наприклад: Internet Explorer, Opera, Mozilla Firefox, Netscape Navigator та ін. Найбільш популярними реалізаціями веб-серверів є: InternetInformation Services (IIS), Apache,

lighttpd, nginx. Найбільш відомі реалізації проксі-серверів: Squid, UserGate, Multiproxy, Naviscope.

"Класична" схема HTTP-сеансу виглядає так.

1. Встановлення TCP-з'єднання.
2. Запит клієнта.
3. Відповідь сервера.
4. Розрив TCP-з'єднання.

Таким чином, клієнт посилає серверу запит, отримує від нього відповідь, після чого взаємодія припиняється. Зазвичай запит клієнта є вимога передати HTML-документ або який-небудь інший ресурс, а відповідь сервера містить код цього ресурсу.

Методи запиту та поля заголовків

До складу HTTP-запиту, переданого клієнтом сервера, входять наступні компоненти:

- Рядок стану (іноді для її позначення використовують також терміни стро¬ка-статус, або рядок запиту).
- Поля заголовка.
- Порожній рядок.
- Тіло запиту.

Рядок стану разом з *полями заголовка* іноді називають також *заголовком запиту.*

1.Рядок стану має наступний формат:

метод_запиту URL_ресурсу *версія_протокола*_HTTP

Розглянемо компоненти рядка стану,

а) Метод запиту.

Метод, вказаний в рядку стану, визначає спосіб впливу на ресурс, URL якого заданий в тому ж рядку. Метод може приймати значення GET, POST, HEAD, PUT, DELETE і т.д. Незважаючи на велику кількість методів, для веб-програміста по-справжньому важливі лише два з них: GET і POST.

• GET. Згідно формального визначення, метод GET призначається для отримання ресурсу з вказаним URL. Отримавши запит GET, сервер повинен прочитати зазначений ресурс і включити код ресурсу до складу відповіді клієнту. Ресурс, URL якого передається в складі запиту, не обов'язково повинен являти собою HTML-сторінку, файл із зображенням або інші дані. URL ресурсу може вказувати на виконуваний код програми, який, при дотриманні певних умов, повинен бути запущений на сервері. У цьому випадку клієнтові повертається не код програми, а дані, згенеровані в процесі її виконання. Незважаючи на те що, за визначенням, метод GET призначений для отримання інформації, він може застосовуватися і в інших цілях. Метод GET цілком підходить для передачі невеликих фрагментів даних на сервер.

• POST. Згідно з тим же формального визначення, основне призначення методу POST - передача даних на сервер. Однак, подібно методу GET, метод POST може застосовуватися по-різному і нерідко використовується для отримання інформації з сервера. Як і у випадку з методом GET, URL, заданий у рядку стану, вказує на конкретний ресурс. Метод POST також може використовуватися для запуску процесу.

• Методи HEAD і PUT є модифікаціями методів GET і POST.

б) Версія протоколу HTTP

Версія протоколу HTTP, як правило, задається в наступному форматі:

HTTP / версія.модіфікація

Поля заголовка, дозволяють уточнювати запит, тобто передавати серверу додаткову інформацію. Поле заголовка має наступний формат:

Ім'я_поля: Значення

Призначення *поля* визначається його *ім'ям*, яке відділяється від значення двокрапкою. Імена деяких найбільш часто зустрічаються в запиті клієнта полів заголовка і їх призначення наведені в таблиці 3.1.

Таблиця 3.1. Поля заголовка запиту HTTP.

Поля заголовка HTTP-запиту	**Значення**
Host	Доменне ім'я або IP-адресу сайту, до якого звертається клієнт
Referer	URL документа, який посилається на ресурс, зазначений у рядку стану
From	Адреса електронної пошти користувача, що працює з клієнтом
Accept	MIME-типи даних, оброблюваних клієнтом. Це поле може мати кілька значень, відокремлюваних одне від іншого запитом. Часто поле заголовка Accept використовується для того, щоб повідомити сервер про те, які типи графічних файлів підтримує клієнт
Accept-Language	Набір однорядкове резюме ідентифікаторів, розділених комами, які позначають мови, які підтримуються клієнтом
Accept-Charset	Перелік підтримуваних наборів символів
Content-Type	MIME-тип даних, що містяться в тілі запиту (якщо клопотання не складається з одного заголовка)
Content-Length	Число символів, що містяться в тілі запиту (якщо клопотання не складається з одного заголовка)
Range	Присутній в тому випадку, якщо клієнт запитує не весь документ, а лише його частина
Connection	Використовується для управління TCP-з'єднанням. Якщо в полі міститься Close, це означає, що після обробки запиту сервер повинен закрити з'єднання. Значення Keep-Alive пропонує не закривати TCP-з'єднання, щоб воно могло бути використано для подальших запитів
User-Agent	Інформація про клієнта

У багатьох випадках при роботі в Веб тіло запиту відсутня. При запуску CGI-сценаріїв дані, передані для них у запиті, можуть розміщуватися в тілі запиту.

Нижче представлений приклад HTML-запиту, згенерованого браузером

GET http://oak.oakland.edu/ HTTP / 1.0

Connection: Keep-Alive
User-Agent: Mozilla / 4.04 [en] (Win95; I)
Host: oak.oakland.edu
*Accept: image / gif, image / x-xbitmap, image / jpeg, image / pjpeg, image / png, * / **
Accept-Language: en
*Accept-Charset: iso-8859-l, *, utf-8*

Отримавши від клієнта запит, сервер повинен відповісти йому. Знання структури відповіді сервера необхідно розробнику веб-додатків, так як програми, які виконуються на сервері, повинні самостійно формувати відповідь клієнту.

Подібно запиту клієнта, відповідь сервера також складається з чотирьох перерахованих нижче компонентів.

- Рядок стану.
- Поля заголовка.
- Порожній рядок.
- Тіло відповіді.

Відповідь сервера клієнту починається з рядка стану, яка має наступний формат:

Версія_протокола _ Код_відповіді _ Пояснювальне повідомлення

Розглянемо складові рядка стану відповіді клієнту:

•*Версія_протокола* задається в тому ж форматі, що і в запиті клієнта, і має той же зміст.

•*Код_відповіді* - це тризначне десяткове число, що представляє в закодованому вигляді результат обслуговування запиту сервером.

•*Пояснювальне повідомлення* -дублює код відповіді в символьному вигляді. Це рядок символів, яка не обробляється клієнтом. Він призначений для системного адміністратора або оператора.

З трьох цифр складових коду відповіді, перша (старша) визначає клас відповіді, інші дві являють собою номер відповіді усередині класу. Так,

наприклад, якщо запит був оброблений успішно, клієнт отримує наступне повідомлення:

HTTP / 1.0 200 OK

Як видно, за версією протоколу HTTP 1.0 слід код 200. У цьому коді символ 2 означає успішну обробку запиту клієнта, а інші дві цифри (00) - номер даного повідомлення.

У використовуваних в даний час реалізаціях протоколу HTTP перша цифра не може бути більше 5 і визначає наступні класи відповідей:

1 - спеціальний клас повідомлень, званих інформаційними. Код відповіді, що починається з 1, означає, що сервер продовжує обробку запиту. При обміні даними між HTTP-клієнтом і HTTP-сервером повідомлення цього класу використовуються досить рідко.

2 - успішна обробка запиту клієнта.

3 - перенаправлення запиту. Щоб запит був обслужений, необхідно пред ¬ вжити додаткових дії.

4 - помилка клієнта. Як правило, код відповіді, що починається з цифри 4, повер - щається в тому випадку, якщо в запиті клієнта зустрілася синтаксична помилка.

5 - помилка сервера. З тих чи інших причин сервер не в змозі ви ¬ повнити запит.

Приклади кодів відповідей, які клієнт може отримати від сервера, і пояснюю ¬ щие повідомлення наведені в таблиці 3.2.

Таблиця 3.2. Класи кодів відповіді сервера.

Код	Розшифровка	Інтерпретація
100	Continue	Частина запиту прийнята, і сервер очікує від клієнта продовження запиту

200	OK	Запит успішно оброблений, і у відповіді клієнта передаються дані, зазначені в запиті
201	Created	В результаті обробки запиту був створений новий ресурс
202	Accepted	Запит прийнятий сервером, але обробка його не закінчено. Даний код відповіді не гарантує, що запит буде оброблений без помилок.
206	Partial Content	Сервер повертає частину ресурсу у відповідь на запит, що містив поле заголовка Range
301	Multiple Choice	Запит вказує більш ніж на один ресурс. У тілі відповіді можуть міститися вказівки на те, як пра¬вільно ідентифікувати запитуваний ресурс
302	Moved Permanently	Викликана ресурс більше не розташовується на сервері
302	Moved Temporarily	Викликана ресурс тимчасово змінив свою адресу
400	Bad Request	У запиті клієнта виявлена синтаксична помилка
403	Forbidden	Наявний на сервері ресурс недоступний для даного користувача
404	Not Found	Ресурс, зазначений клієнтом, на сервері відсутній
405	Method Not Allowed	Сервер не підтримує метод, зазначений у запиті
500	Internal Server Error	Один з компонентів сервера працює некоректно
501	Not Implemented	Функціональних можливостей сервера недоста¬точно, щоб виконати запит клієнта
503	Service Unavailable	Служба тимчасово недоступна
505	HTTP Version not Supported	Версія HTTP, зазначена в запиті, що не поддержі¬вается сервером

У відповіді використовується така ж структура полів з аголовка, як і в запиті клиен та. Поля заголовка призначені для того, щоб уточнити відповідь сервера клієнту. Опис деяких полів, які можна зустріти у заголовку відповіді сервера, наведено в таблиці 3.3.

Таблиця 3.3. Поля заголовка відповіді веб- сервера

Ім'я поля	Опис вмісту
Server	Ім'я та номер версії сервера
Age	Час у секундах, що минув з моменту створення ресурсу
Allow	Список методів, допустимих для даного ресурсу
Content-Language	Мови, які повинен підтримувати клієнт для того, щоб коректно відобразити переданий ресурс
Content-Type	*MIME-тип* даних, що містяться в тілі відповіді сервера
Content-Length	Число символів, що містяться в тілі відповіді сервера
Last-Modified	Дата і час останньої зміни ресурсу
Date	Дата і час, що визначають момент генерації відповіді
Expires	Дата і час, що визначають момент, після якого інформація, передана клієнту, вважається застарілою
Location	У цьому полі вказується реальне розташування ресурсу. Воно використовується для перенаправлення запиту
Cache-Control	Директиви управління кешуванням. Наприклад, *no-cache* означає, що дані не повинні кешуватися

У тілі відповіді міститься код ресурсу, переданого клієнтові у відповідь на запит. Це не обов'язково має бути HTML-текст веб-сторінки. У складі відповіді можуть передаватися зображення, аудіо-файл, фрагмент відеоінформації, а також будь-який інший тип даних, підтримуваних клієнтом.

Специфікація MIME (Multipurpose Internet Mail Extension - багатоцільове поштове розширення Internet) спочатку була розроблена для того, щоб забезпечити передачу різних форматів даних у складі електронних листів. Однак застосування MIME не вичерпується електронною поштою. Засоби MIME успішно використовуються в WWW і, по суті, стали невід'ємною частиною цієї системи.

Стандарт MIME розроблений як розширювана специфікація, в якій передбачається, що число типів даних буде рости в міру розвитку форм представлення даних. Кожен новий тип в обов'язковому порядку повинен бути зареєстрований в IANA (Internet Assigned Numbers Authority).

До появи MIME комп'ютери, які взаємодіють по протоколу HTTP, обмінювалися виключно текстовою інформацією. Для передачі зображень, як і для передачі будь-яких інших довічних файлів, доводилося користуватися протоколом FTP.

Відповідно до специфікації MIME, для опису формату даних використовуються тип і підтип. Тип визначає, до якого класу належить формат вмісту HTTP-запиту або HTTP-відповіді. Підтип уточнює формат. Тип і підтип відокремлюються друг від друга косою рисою:

тип / підтип

Оскільки в переважній більшості випадків у відповідь на запит клієнта сервер повертає початковий текст HTML-документа, то в поле Content-type відповіді зазвичай міститься значення *text/html.* Тут ідентифікатор *text* визначає тип, повідомляючи, що клієнту передається символьна інформація, а ідентифікатор *html* описує підтип, тобто вказує на те, що послідовність символів, що міститься в тілі відповіді, являє собою опис документа на мові HTML.

Перелік типів і підтипів MIME досить великий. У таблиці 3.4 наведені приклади MIME-типу, найбільш часто зустрічаються в заголовках HTML-запитів і відповідей.

Таблиця 3.4. MIME типи даних.

Тип / підтип	**Розширення файлу**	**Опис**
application / pdf	.pdf	Документ, призначений для обробки Acrobat Reader

application / msexcel	.xls	Документ у форматі Microsoft Excel
application / postscript	.ps, .eps	Документ в форматі PostScript
application / x-tex	.tex	Документ у форматі TeX
application / msword	.doc	Документ у форматі Microsoft Word
application / rtf	.rtf	Документ у форматі RTF, який відображається за допомогою Microsoft Word
image / gif	.gif	Зображення в форматі GIF
image / jpeg	.jpeg, .jpg,	Зображення у форматі JPEG
image / tiff	.tiff, .tif	Зображення у форматі TIFF
image / x-xbitmap	.xbm	Зображення у форматі XBitmap
text / plain	.txt	ASCII- текст
text / html	. html,. htm	Документ у форматі HTML
audio / midi	.midi, .mid	Аудіофайл у форматі MIDI
audio / x-wav	.wav	Аудіофайл у форматі WAV
message / rfc822		Поштове повідомлення
message / news		Повідомлення в групи новин
video / mpeg	.mpeg, .mpg, .mpe	Відеофрагмент у форматі MPEG
video / avi	.avi	Відеофрагмент у форматі AVI

Механізм Cookie

Оскільки HTTP-сервер не пам'ятає передісторії запитів клієнтів, то кожний запит обробляється незалежно від інших, і у сервера немає можливості визначити, чи виходять запити від одного клієнта або різних клієнтів.

Якщо сервер перевірятиме TCP-з'єднання і запам'ятовувати IP-адреси комп'ютерів-клієнтів, він все одно не зможе розрізнити запити від двох браузерів, що виконуються на одній машині. І навіть якщо припустити, що на комп'ютері працює лише одна клієнт-програма, то ніхто не може стверджувати, що в проміжку між двома запитами вона не була завершена, а потім запущена знову вже іншим користувачем.

Якщо ви користувалися поштовою скринькою на mail.ukr.net або на іншому сервері, що надає поштові послуги користувачам, Веб згадайте, як поводився клієнт після того, як ви створили для себе поштову скриньку на сервері. Коли ви наступного разу звернулися з того ж комп'ютера до mail.ukr.net, ви, мабуть, помітили, що після завантаження веб-сторінки ваше реєстраційне ім'я вже відображалося у відповідному полі введення.

Такі відомості дозволяє отримати додатковий засіб під назвою cookie . *Механізм cookie дозволяє серверу зберігати інформацію на комп'ютері клієнта і витягувати її звідти.*

Ініціатором записи cookie виступає сервер. Якщо у відповіді сервера п рісутствует поле заголовка Set-cookie , клієнт сприймає це як команду на запис cookie . Надалі, якщо клієнт звертається до сервера, від якого він раніше прийняв поле Set- cookie , крім іншої інформації він передає серверу дані cookie . Для передачі зазначеної інформації сервера використовується поле заголовка Cookie.

Для того щоб в загальних рисах уявити собі, як відбувається обмін даними cookie , розглянемо наступний приклад. Припустимо, що клієнт передає запити на сервери A , B і C . Припустимо також, що сервер B , на відміну від A і C , передає клієнту команду записати cookie . Послідовність запитів клієнта сервера і відповідей на них буде виглядати приблизно наступним чином:

1. Передача запиту серверу A .
2. Отримання відповіді від сервера A .

3. Передача запиту серверу B .

4. Отримання відповіді від сервера B . До складу відповіді входить поле заголовка SetCookie . Отримавши його, клієнт записує cookie на диск.

5. Передача запиту серверу C . Незважаючи на те, що на диску зберігається запис cookie , клієнт не робить ніяких спеціальних дій, так як значення cookie було записано з ініціативи іншого сервера.

6. Отримання відповіді від сервера З .

7. Передача запиту серверу A . У цьому випадку клієнт також ніяк не реагує на той факт, що на диску зберігається cookie .

8. Отримання відповіді від сервера A .

9. Передача запиту серверу B . Перед тим як зформувати запит, клієнт визнаделяет, що на диску зберігається запис cookie , створений після отримання відповіді від сервера B . Клієнт перевіряє, чи задовольняє даний запит деяким вимогам, і, якщо перевірка дає позитивний результат, включає в поле Cookie .

Таким чином, процедуру запису та отримання cookie можна уявити собі як своєрідний "запит" серверу, інкапсульований в його відповіді клієнту. Отримання cookie також можна представити як відповідь клієнта, інкапсулюваної у складі запиту того ж сервера.

Розглянемо докладніше, які дані передаються в поле заголовка Set-cookie і як вони впливають на поведінку клієнта.

Поле Set- cookie має наступний формат:

ім'я = значення; expires = дата; path = шлях;домен=ім'я_домена, secure

де:

- Пара *ім'я = значення* - іменовані дані, які зберігаються за допомогою механізм cookie . Ці дані повинні зберігатися на клієнт-машині і передаватися сервера у складі чергового запиту клієнта.
- *дата*, яка є значенням параметра *expires* , визначає час, після закінчення якого інформація cookie втрачає свою актуальність. Якщо

ключове слово expires відсутнє, дані cookie видаляються після закінчення поточного сеансу роботи браузера.

- Значення параметра *domain* визначає домен, з яким пов'язуються дані cookie . Щоб дізнатися, чи слід передавати у складі запиту дані cookie, браузер порівнює доменне ім'я сервера, до якого він збирається звернутися, з доменами, які пов'язані із записами cookie , що зберігаються на клієнт-машині. Результат перевірки буде вважатися позитивним, якщо сервер, якому направляється запит, належить домену, пов'язаному з cookie . Якщо відповідність не виявлено, дані cookie не передаються.
- Шлях, зазначений у якості значення параметра *path* , дозволяє виконати подальшу перевірку і прийняти остаточне рішення про те, чи слід передавати дані cookie у складі запиту. Крім домену із записом cookie зв'язується шлях. Якщо браузер виявив відповідність імені домена значенням параметра domain , він перевіряє, чи відповідає шлях до ресурсу шляху, пов'язаному з cookie . Порівняння вважається успішним, якщо ресурс міститься в каталозі, вказаному допомогою ключового слова *path* , або в одному з його підкаталогів. Якщо і ця перевірка дає позитивний результат, дані cookie передаються сервера. Якщо параметр *path* в поле Set - Cookie відсутній, то вважається, що записcookie пов'язана з URL конкретного ресурсу, переданого сервером клієнтові.
- Останній параметр, *secure* , вказує на те, що дані cookie повинні передаватися по захищеному каналу.

Для передачі даних cookie сервера використовується поле заголовка Cookie

Формат цього поля досить простий ой:

Cookie : ім'я = значення; ім'я = значення ; ...

За допомогою поля Cookie передається одна або декілька пар і*м'я = значення*. Кожна з цих пар належить записи cookie , для якої URL

запитуваного ресурсу відповідають імені домена і шляхи, зазначеним раніше в поле Set-cookie .

Проксі-сервер

За часів брандмауерів, VPN-сервісів і додатків Tor, практичних методів маскування IP-адрес і прискорення Інтернету, проксі-сервер стає незамінний для активно серфінгу в Інтернеті.

Термін проксі походить від латинського «proxus» і означає «наступний» або «замінник». Він являє собою мережевий інтерфейс зв'язку, який забезпечує комутацію між двома віддаленими машинами.

Існує два типи проксі:

1. Перші - служать кешем, тобто гарантують, що дані не потрібно перезавантажувати з Інтернету. Вони працюють локально в мережі - це має сенс в компаніях, наприклад, для полегшення передачі по корпоративним інтернет-ліням.
2. Другі - надаються провайдером. Проксі-сервер провайдера особливо цікавий для повсякденного приватного використання: він не тільки прискорює мережевий доступ, але також може забезпечити вашу повну анонімність. Це означає, що сервіс провайдера може дати значно більшу швидкість та захист, і легко налаштовується.

Головна мета будь-якого проксі-сервера - забезпечувати прийом запитів клієнта від імені сервера і перенаправляти трафік на цільовий комп'ютер з окремим IP-ідентифікатором. В цьому випадку пряма комутація між вашим комп'ютером і веб-ресурсом або сервером відсутня. Сервер отримує дані, а потім перенаправляє їх одержувачу. Це дозволяє, наприклад, встановити комунікаційне з'єднання між двома пристроями, якщо вони фактично несумісні один з одним.

Всі типи проксі-сервери зазвичай можна класифікувати наступним чином**:**

- Прямі - категорія проксі за замовчуванням, в даному випадку він служить посередником між комп'ютером і ширшої глобальною мережею.
- Зворотні - Цей тип серверів діє як посередник між Інтернетом і невеликою групою аналогічних пристроїв. Наприклад, зворотний проксі-сервер здатний служити концентратором-шлюзом між Інтернетом і локальною мережею компанії.
- Відкриті - будь-який користувач онлайн може отримати доступ до відкритих проксі (також відомим як «публічні проксі»). Вони зазвичай пропонуються публіці безкоштовно і забезпечують досить широкий функціонал.

В епоху, коли веб-ресурси і онлайн-сервіси збирають величезну кількість приватної інформації - будь то кількісні дані, наприклад, «де відвідувачі мого сайту найчастіше натискають», або конкретні, відчутні дані, такі як хобі, уподобання і статистика пошуку, - проблема захисту даних і конфіденційність стає все більш актуальною темою.

Проксі-сервери можуть кешувати дані. Як тільки ви отримуєте доступ до певних веб-сайтів, ви зберігаєте їх для подальшого використання. Коли запитується кешована сторінка, вона може відображатися користувачеві швидше.

Ще одним значною перевагою проксі-сервера є те, що адміністратор може заборонити користувачам доступ до веб-сайтів, які можуть бути небезпечними, тобто проксі - сервери можуть робити фільтрацію шкідливих сайтів.

Більшість інтернет-провайдерів пропонують потужні проксі-сервери, наприклад, такий який підтримує сервіс хостингу ukraine.com.ua. Однак існує і безліч публічних сервісів, але ставитися до них треба з обережністю і раз в декілька підключень перевіряти дані.

Можна завантажити списки серверів з так званих проксі-листів. Ці списки є у вільному доступі. Після цього слід вручну занести адресу сервера в браузер і службу проксі вашої ОС, застосувавши інструкцію.

Дані, які перенаправляються через проксі-сервер, можуть фільтруватися по параметрам, проходити буферизацию і перерозподілятися по інших серверах для розподілу навантаження. Крім цього, проксі-сервер є головним елементом брандмауера, який охороняє ваш ПК від кібератак з глобальної мережі.

Системні проксі використовуються всіма програмами, включаючи всі браузери, такі як Chrome, Edge або Firefox. Однак проксі в браузері використовується тільки там, де він встановлений. Це дає декілька переваг: наприклад, ви можете управляти браузером виключно з проксі, але запускати всі інші інтернет-додатки без проксі. Або, навпаки, ви можете обробляти весь інтернет-трафік через проксі.

Браузер Internet Explorer, а також Google Chrome і Opera автоматично дублюють дані встановленої на комп'ютері операційної системи, а Mozilla Firefox вимагає введення окремих установок служби проксі, яка відноситься тільки до використання цього браузера.

Обираючи проксі треба бути надзвичайно обережним у виборі провайдера. Власник проксі-сервера може контролювати вашу активність, якщо ви отримуєте онлайн доступ до інформації через HTTP замість HTTPS. Адміністратор служби може заблокувати безпечні ресурси без пояснення причин, якщо він цього захоче. Деякі провайдери обмежують пропускну здатність сервісів.

Провайдери часто застосовують особливі «прозорі» проксі, а встановлене на них ПО здатне передавати ресурсу інформацію про вашу IP-ідентифікатор. Ці служби в першу чергу служать для підвищення швидкості завантаження даних і не повинні використовуватися для маскування IP-адреси.

З мінусів використання проксі-серверів можна азначити, що80% безкоштовних служб не використовують протокол HTTPS з SSL-протоколом, тому ви практично не захищені від кібератак онлайн. А деякі з них можуть змінювати статичну розмітку HTML і використовувати змінений код JavaScript для передачі реклами своїм відвідувачам.

Треба обирати приватні проксі перевірених провайдерів. Вони надають додатковий рівень безпеки, використовуючи службу VPN, яка забезпечує доступ до проксі-службі. Якщо ви не впевнені, чому вам слід застосовувати VPN замість проксі-сервера, врахуйте наступне: VPN вже пропонує ті ж переваги, що і анонимайзер, а також задіє надійні протоколи шифрування. Це забезпечує більш високий рівень безпеки в серфінгу в Інтернеті і захист вашої конфіденційної інформації.

3.2. Клієнтські сценарії та програми

3.2.1 Взаімодія браузера з веб-сервером

Веб-додаток - додаток, це додаток , в якому клієнтом виступає браузер, а сервером - веб-сервер.

Розглянемо типи програм, що забезпечують роботу Веб і використовують HTTP-протокол.

Ніякій HTTP-обмін неможливий без клієнта і сервера. Однак, крім клієнта і сервера в веб - сеансе можуть брати участь і інші програми, які і є об'єктом веб-програмування.

Результатом роботи веб-додатки є веб-сторінка, яка відображається у вікні браузера. При цьому саме веб-додаток може виконуватися як на комп'ютері клієнта, так і на комп'ютері сервера.

Розглянемо докладніше обидві схеми.

1. *Програми, що виконуються на клієнт-машині*

Одним з типів програм, призначених для виконання на клієнт-машині, є сценарій наприклад, JavaScript (VBScript). Оригінальний текст сценарію

являє собою частину веб-сторінки, тому сценарій JavaScript передається клієнту разом з документом, до складу якого він входить. Обробляючи HTML-документ, браузер виявляє вихідний текст сценарію і запускає його на виконання.

До всіх програм, які передаються з сервера на клієнт-машини і запускаються на виконання, пред'являється одна загальна вимога: *ці програми повинні бути позбавлені можливості звертатися до ресурсів комп'ютера, на якому вони викоуються.* Така вимога цілком обгрунтовано. Адже передача по мережі і запуск Java- апплетів і JavaScript-сценаріїв відбувається автоматично без участі користувача, тому робота цих програм повинна бути абсолютно безпечною для комп'ютера. Іншими словами, мови, призначені для створення програм, що виконуються на клієнт-машині, повинні бути абсолютно непридатні для написання вірусів і подібних програм.

2. *Програми, що виконуються на сервері*

Код програми, що працює на сервері, не передається клієнту. При отриманні від клієнта спеціального запиту, який передбачає виконання такої програми, сервер запускає її і передає параметри, що входять до складу запиту. Засоби для генерації подібного запиту зазвичай входять до складу HTML-документа.

Результати своєї роботи програма оформляє у вигляді HTML-документа і передає їх веб-сервер, а останній, у свою чергу, доповнює отримані дані HTTP-заголовком і передає їх клієнту. Взаємодію клиента і сервера в цьому випадку показано на рисунку 3.2.1.

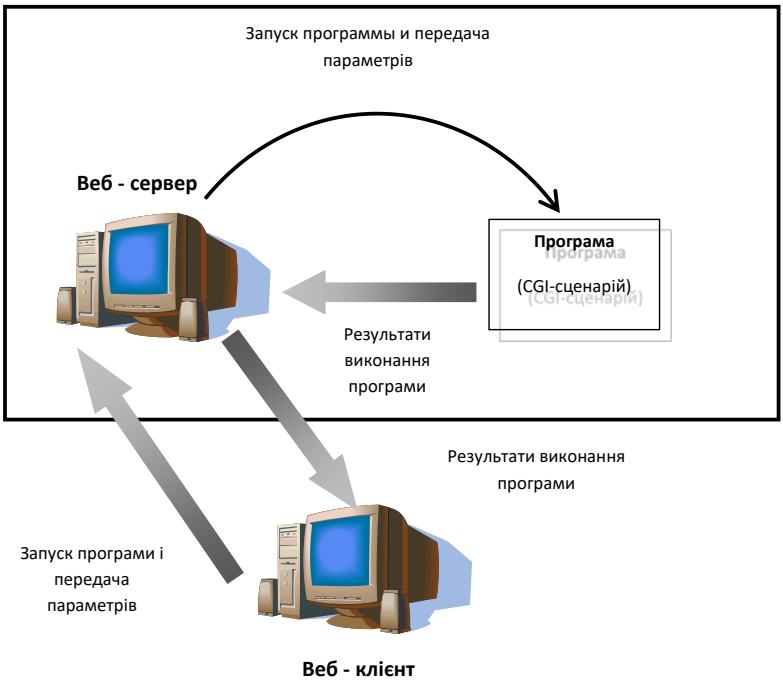


Рисунок 3.2.1 - Взаємодія клієнта з програмою, що виконується на сервері.

3. Насичені інтернет-додатки

Насичене інтернет-додаток (Rich Internet application) - ще один підхід, який полягає у використанні Adobe Flash або Java-аплетів для повної або часткової реалізації інтерфейсу користувача, оскільки більшість браузерів підтримує ці технології (як правило, за допомогою плагінів).

Виникнення даного підходу обумовлена тим, що в рамках веб-додатків з "тонким" клієнтом взаємодія користувача з додатком реалізується в істотному ступені через сервер, що вимагає відправки даних на сервер, отримання відповіді від сервера і перезавантаження сторінки на стороні клієнта.

При використанні Java-аплетів до складу HTML-документа включається спеціальний дескриптор, що описує розташування файлу, що містить код

аплета на сервері. Після того, як клієнт отримує HTML-код документа, що включає аплет, він генерує додатковий запит серверу. Після того, як сервер пересилає клієнту код аплета, сам аплет запускається на виконання. Взаємодія між клієнтом і сервером при отриманні аплета показано на рисунку 3.2.2.

Для розробки насичених інтернет-додатків використовуються пакети Curl, Adobe Flex і Microsoft Silverlight.

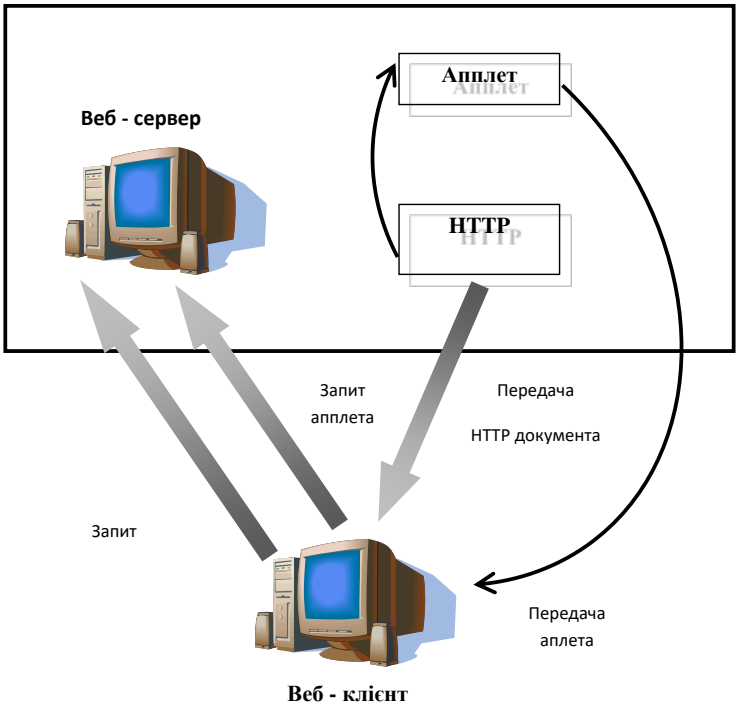


Рисунок 3.2.2-.Передача клієнту Java - аплета.

Для розробки насичених інтернет-додатків використовуються пакети Curl, Adobe Flex і Microsoft Silverlight.

Java-аплети

Java-аплет - це програма, написана на мові Java і відкомпілюватнав байт-код, виконується в браузері з використанням віртуальної Java-машини (JVM). Аплети використовуються для надання інтерактивних можливостей веб-додатків, які не можливі в HTML. Так як байт-код Java платформо-незалежний, то Java-аплети можуть виконуватися браузерами на багатьох операційних платформах.

Java-сервлети є серверними додатками, але вони відрізняються від аплетів мовою, функціями та іншими характеристиками.

Призначення Java-аплети –виконання їх в безпечному середовищі з метою запобігання доступу до локальних ресурсів комп'ютера клієнта.

Код аплету завантажується з веб-сервера, і браузер

- або вставляє апплет у веб-сторінку;
- або відкриває окреме вікно з власним користувача інтерфейсом аплету.

Аплет може бути впроваджений у веб-сторінку за допомогою використання HTML тега <applet>, або (що рекомендується) тега <object>.

Можна назвати наступні переваги Java-аплетів:

- працюють практично на більшості операційних платформ;
- підтримуються більшістю браузерів;
- кешируются в більшості браузерів, що істотно прискорює їх завантаження при поверненні на веб-сторінку;
- після першого запуску аплета, коли Java-машина вже виконується і швидко запускається, виконання аплетів відбувається значно швидше;
- завантажуються зі швидкістю порівнянної з програмами на інших компільованих мовах, наприклад C ++, але у багато разів швидше ніж на JavaScript.

При цьому у Java-аплетів є й недоліки :

- потрібна установка Java-розширення, які доступні за замовчуванням не у всіх браузерах;
- проблеми реалізації Java-розширень для 64-розрядних процесорів;
- не можуть запускатися до першого завантаження віртуальної Java-машина, що може займати значний час;
- розробка користувальницького інтерфейсу з використанням аплетів є більш складним завданням в порівнянні з HTML;
- не мають прямого доступу до локальних ресурсів комп'ютера клієнта; деякі аплети прив'язані до використання певного середовища часу виконання Java (JRE).

3.2.2 Коротка характеристика мов програмування: JavaScript, JScript, VBScript

Для інтерактивної та активної взаємодії користувача з web-додатком використовується мова програмування java script.

Java script — це динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду виконуватися на стороні клієнта, таким чином знімаючи навантаження з основного сервера на якому і виконується основний функціонал системи. Ця мова також може використовується для програмування на стороні сервера, розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному, всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як

об'єкти першого класу . Враховуючи виявлені переваги та недоліки проаналізованих засобів для створення webзастосунків призначених для ведення статистики, найкраще підходять такі технології: YII2, Java Script, HTML5 та CSS3

JavaScript - це об'єктно –орієнтована мова програмування сценаріїв з синтаксисом, який має аналоги з синтаксисом мов C, Perl, Pyton. Щоб веб-сторінка була інтерактивною і динамічною, необхідно використовувати скрипти, або сценарії.

Сценарій (script, скрипт) — це програма, написана спеціальною мовою програмування і вбудована в HTML-документ. Сценарії описують усі можливі дії над елементами HTML-документа під час взаємодії з користувачем (наприклад, реакцію на натискання кнопки миші, зміну вмісту сторінки залежно від певних дій користувача тощо). Сценарії вбудовуються в HTML-документ трьома стандартними засобами:

- у вигляді гіперпосилання;
- у вигляді обробника подій;
- у контейнерний тег <script>.

За допомогою веб-сценаріїв можна створити принципово новий інтерфейс користувача для своєї сторінки. Всі події, генеровані браузером, такі як клацання кнопок, модифікація полів форм і переміщення між сторінками, можна перехопити й обробити засобами JavaScript.

Основні області застосування мови JavaScript:

- динамічне створення документа HTML за допомогою скриптів;
- перевірка достовірності полів форм HTML до передавання їх на сервер;
- локальне введення інформації для керування програмою;
- надання користувачу можливості вибору операцій, виконуваних браузером;

- виведення повідомлень для користувача у діалогових вікнах;
- локальне опрацювання форм, введення інформації користувачем.
- посилання запиту на сервер і завантажувати дані без перезавантаження сторінки (ця технологія називається "AJAX").

Браузер інтерпретує програму, створену мовою JavaScript, під час завантаження документа, в який вміщено її код. Проте різні браузери сприймають різні її варіанти. Версія мови JavaScript від корпорації Майкрософт, що має назву JScript, є найближчою до стандарту. Браузер Microsoft Internet Explorer підтримує не лише JScript, а й ще одну мову скриптів — Visual Basic Script (VBScript).

Щоб використовувати мову скриптів ефективно, необхідно орієнтуватися в об'єктній моделі HTML-документа.

Є три *чудові особливості JavaScript***:**

- Повна інтеграція з HTML/CSS.
- Прості речі робляться просто.
- Підтримується всіма поширеними браузерами і включений за умовчанням.

Цих трьох речей одночасно немає більше ні в одній браузерній технології. Тому JavaScript і є самим поширеним інтерфейсом. Іноді кросс-браузерна розробка стає непростою справою. Саме тому браузери поділяють по мірі підтримки.

Наприклад :

1. Останні версії Firefox, Internet Explorer, Safari/Chrome

   Ідеальна підтримка.

2. Opera та попередні версії сучасних браузеров

   Можливі незначні «помарки», які не ламають функціонал.

3. Старі браузери

   Підтримують тільки базові можливості.

4. Дуже старі браузери, текстові браузери

Не підтримуються.

JScript

Синтаксис JScript багато в чому аналогічний мови JavaScript, однак, крім додавання клієнтських скриптів на веб-сторінки і деяких інших функцій, JScript може використовуватися і для інших цілей, наприклад:

- автоматизація адміністрування систем Microsoft Windows;
- створення сторінок ASP.

Мова JScript отримав подальший розвиток у вигляді мови JScript.NET, який орієнтований на роботу в рамках платформи Microsoft.NET

JScript - інтерпретована, об'єктно-орієнтована мова. Хоча має істотно меншу кількість можливостей, ніж такі об'єктно-орієнтовані мови як C ++ і Java.

Можливості мови істотно обмежені:

- мова не дозволяє розробляти окремі програми;
- сценарії на JScript можуть виконуватися тільки за допомогою інтерпретатора, зокрема веб-браузер.
- JScript - мова без суворого контролю типів. Тому не потрібно оголошувати тип змінних явно. Крім того, в про багатьох випадках JScript виконує перетворення автоматично, коли вони необхідні. Наприклад, при додаванні рядка і числа, число буде перетворено в рядок.

Код на JScript пишеться в текстовому форматі, і організований в інструкції, блоки, що складаються з пов'язаних наборів інструкцій, і коментарів. В межах інструкції можна використовувати змінні і дані, такі як рядки, числа і вирази. Для оголошення кінця інструкції крапку з комою (;). Група JScript-інструкцій, укладена у фігурні дужки {}, називаєтьсяблоком.

JScript - мову з нестрогим контролем типів, змінні в JScript не мають строго фіксованого типу. Змінні мають тип, еквівалентний типу значення,

яке вони містять. Проте, в деяких випадках, необхідно примусове перетворення змінної в певний тип. Числа можуть бути оголошені як рядки, а рядки необхідно перетворити в числовий тип. Для цього застосовують функції parseInt () і parseFloat ().

В JScript використовується шість типів даних. Основні з них - числа, рядки, об'єкти, логічний. Інші два - null і undefined (тобто невизначений).

Мова підтримує умовні вираження if і if ... else. При використанні декількох умов одночасно можна використовувати оператори || (АБО) або && (І).

В JScript підтримується кілька типів циклів: for, for ... in, while, do ... while і switch. Також існує інструкція зупинення виконання циклу. Оператор завершення break може використовуватися, щоб зупинити цикл, при виконанні якого-небудь умови. Інструкція continue використовується, щоб негайно перейти до виконання наступної ітерації, пропускаючи іншу частину виконання коду поточної ітерації, але оновлюючи змінну-лічильник.

В JScript є два види функцій: вбудовані і зумовлені. Програміст має можливість створювати власні функції. Визначення функції складається з оголошення параметрів і блоку інструкцій JScript.

Всі об'єкти в JScript можна розділити на три види: вбудовані , створені і браузерні . Про бработка об'єктів і масивівідентична. Можна звернутися до будь-якої частини об'єкта (його властивостей і методів) або по імені, або за індексом. Нумерація індексів в JScript починається з нуля.

Коротка характеристика VBScript

Visual Basic Scripting Edition (зазвичай просто VBScript) - сценарна мова програмування, що інтерпретується компонентом Windows Script Host. Вона широко використовується при створенні скриптів в операційних системах сімейства Microsoft Windows.

Мова був створений компанією Microsoft як заміна застарілого пакетного мови, інтерпретованих додатком command.com. Синтаксис VBScript є спрощеною версією синтаксису мови Visual Basic.

Сценарії мовою VBScript найчастіше використовуються в наступних областях, що використовують програмні продукти Microsoft:

- автоматизація адміністрування систем Windows;
- серверний програмний код в сторінках ASP;
- клієнтські сценарії в браузері Internet Explorer.

ActionScript - загальна характеристика.

ActionScript - об'єктно-орієнтована мова програмування, один з діалектів EcmaScript, який додає інтерактивність, обробку даних та багато іншого у вміст Flash-додатків. ActionScript виповнюється віртуальною машиною (ActionScript Virtual Machine), яка є складовою частиною додатка Flash Player. ActionScript компілюється в байткод, який включається в SWF-файл.

SWF-файли виконуються Flash Player. Сам Flash Player існує у вигляді плагіна до веб-браузера, а також як самостійне виконувався додаток. У другому випадку можливе створення виконуваних exe-файлів, коли swf-файл включається у Flash Player.

За допомогою ActionScript можна створювати інтерактивні мультимедіа-додатки, ігри, веб-сайти та багато іншого.

3.2.3. Мови розробки сценаріїв. PHP, PERL, PYTHON

Однією з перших технологій створення веб-застосувань, які виконуються сервером, була Common Gateway Interface (CGI) технологія .

Вона дозволила розробку і виконання серверних застосувань, звернення до яких відбувається за допомогою зазначеного в URL імені (та параметрів). Залежно від обраного протоколу вхідною інформацією таких

веб-додатків вважають безпосередньо код HTTP-заголовка або запит пошукової системи.

CGI-застосування − це консольні додатки, які генерують HTML-код, переданий браузеру. Серед інших популярних технологій, які реалізують створення веб - сторінок із фрагментами коду, виконуваного на сервері, виділимо некомерційну, вільно розповсюджувану технологію PHP (Personal Home Pages) .Ця технологія заснована на використанні CGI-застосувань, що інтерпретують впроваджений у HTML-сторінку код на скриптовій мові.

PHP

Створювалась мова PHP як засоби генерації HTML-сторінок на стороні веб-сервера. PHP-це мова програмування, код якої вбудовується безпосередньо в HTML-сторінку.

Програму, написану на PHP, називають PHP-скриптом. При запиті користувача web-сервер переглядає документ, виконує знайдені в ньому PHP-інструкції (оператори мови PHP), а результат їхнього виконання повертає користувачеві. При цьому статична частина документа, написана мовою HTML, фактично є шаблоном, а змінювана частина формується при виконанні PHP-інструкцій і результат повертається програмі-браузеру.

Для віддаленого користувача подібні документи нічим не відрізняються від звичайних статичних HTML-документів, за винятком того, що в розширені імені файлу для таких документів може стояти не htm або html, а php (а в старих версіях phtml або php3).

PHP - це система розробки скриптів, що включає в себе CGI - інтерфейс, інтерпретатор мови та набір функцій для доступу до баз даних і різних об'єктів WWW (функції для роботи з електронною поштою, FTP, http, тощо). На сьогодні, PHP є одним із найбільш зручних і водночас достатньо потужним засобом розробки додатків WWW і інтерфейсів до баз даних в мережі Інтернет.

До переваг також можна віднести:

- наявність інтерфейсів до багатьох баз даних;
- в PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase;
- через стандарт відкритого інтерфейсу зв'язку з базами даних (Open Database Connectivity Standard — ODBC) можна підключатися до всіх баз даних, до яких існує драйвер;
- Підтримка широкого спектру мережевих протоколів (LDAP, FTP, HTTP, IMAP, …) ;
- Работа с різними форматами(Графика, XML, PDF, XLS(X), RSS, HTML, Flash SWF, …);
- Переносимый код Для всех серверных O/C (Linux, Unix, Microsoft Windows, Mac OS X, …) ;
- традиційність.

Завдяки запозиченням під час створення PHP з багатьох інших популярних мов, таких як C, Perl або Pascal, мова PHP здається знайомою програмістам, що працюють в різних областях. Це помітно знижує початкові зусилля при вивченні PHP. Поєднавши в собі переваги інших мов програмування та обравши для себе єдине основне спрямування, PHP стала універсальним, зрозумілим та простим для вивчення засобом роботи в Інтернеті.

Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте, щоб не робили розробники PHP, виконувані файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP цілком достатня для створення цілком серйозних веб-додатків.

Головною особливістю мови PHP є її практичність. PHP надає програмісту інструмент для швидкого й ефективного вирішення поставлених завдань. Вона вирізняється винятковою гнучкістю до потреб розробника. Хоча PHP традиційно рекомендують використовувати у поєднанні з HTML-кодом, проте PHP з таким же успіхом інтегрується і в JavaScript, WML, XML та інші мови Інтернет-програмування [3].

У галузі веб-програмування, особливо в серверному компоненті, PHP є однією з популярних мов сценаріїв (поряд з Perl, JSP та мовами, що використовуються в ASP.NET). Популярність у галузі розробки веб-сайтів виправдовується наявністю великого набору вбудованих інструментів для розробки веб-додатків. Основними з них є:

- Автоматичне вилучення параметрів POST та GET, а також змінних середовища веб-сервера у заздалегідь визначені масиви;
- Робота з авторизацією HTTP;
- Автоматизована відправка заголовків HTTP;
- Взаємодія з великою кількістю різних систем управління базами даних (MySQLi, MySQL, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, SESAM, ODBC, mSQL, Apache Derby та IBM DB2, Informix, Lotus Notes, Cloudscape, dBase, DB ++, DBM, DBX, FrontBase, FilePro, Ingres II, Ovrimos SQL, Firebird / InterBase, Paradox File Access, MaxDB);
- Робота з файлами cookie та сеансами;
- Обробка файлів, завантажених на сервер;
- Робота з локальними та віддаленими файлами, сокетами;
- Робота з XForms.

Незважаючи на те, що PHP не дуже поширений у галузі побудови програм інтерфейсу користувача, його можна використовувати для створення кроссплатформенних додатків за допомогою пакетів PHP-Qt та PHP-GTK, які є обгортками для відповідних популярних бібліотек

графічних елементів та віджетів. Для створення графічних додатків платформи Windows існує безкоштовний пакет WinBinder, який написаний на C і практично використовує виклики до WinAPI. Існують також реалізації PHP для JVM - JPHP і for .NET / Mono - Phalanger. JPHP підтримує розширення Swing, яке майже повністю інтегровано із середовища Java, результатом компілятора коду PHP у Phalanger може бути будь-яка програма .NET.

Недоліками PHP є:

- відсутність підтримки багатопоточності, що не дозволяє розробляти програми для швидкої обробки даних;
- неортогональність і непослідовний синтаксис функцій, що уповільнює розробку додатків;
- Відсутність зворотної сумісності між версіями мови, що ускладнює перехід на нову версію мови та підтримку реалізованих продуктів

Головними перевагами PHP вбачаємо практичність, легкість у застосуванні, ефективність, продуктивність, гнучкість та підтримка переважною більшістю хостинг-провайдерів, а також те, що PHP є проектом відкритого програмного забезпечення.

Отже, підсумовуючи, переваги PHP:

– PHP створювався спеціально для WEB.
– PHP працює швидше “чистых” інтерпретаторів.
– Код PHP обробляється сервером до передачі сторінки браузеру.
– Код PHP може бути безпосередньо вбудован в HTML-код.
– PHP безкоштовній та простий для вивчення.
– Підтримує велику кількість БД.

PHP фреймворки за останній час набрали популярність і стали базовою платформою для розробки веб – застосунків . Використання цих систем, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбавляючи від проблеми

повторюваного коду, і швидко створювати якісні додатки. Між тим, використання PHP фреймворків робить процес створення програми значно більш легким і функціональним.

На сьогоднішній день розроблена величезна кількість PHP фреймворків, кожний з яких заточений під використанні у розробці сайтів певного типу, до найпопулярніших відносяться : ZendFramework, CakePHP, CodeIgniter 2, Kohana 3, Symfony 3,Yii 2.

Python

Python - це високорівнева мова програмування загального призначення, орієнтована на поліпшення читабельності коду та продуктивності розробників. Основний синтаксис Python - мінімалістичний. У той же час, стандартна бібліотека включає велику кількість корисних функцій під час розробки.

Python підтримує кілька парадигм програмування : структуровану, об'єктно-орієнтовану, імперативну, аспектно-орієнтовану та функціональну. Основними архітектурними особливостями є такі функції, як автоматичне управління пам’яттю, динамічне введення тексту, механізм обробки винятків, повний самоаналіз, гнучкі високорівневі структури даних та підтримка багатопотокових обчислень. Код на Python організований у класи та функції, які можна об'єднати в модулі.

Довідковою реалізацією Python є інтерпретатор CPython, який підтримує більшість часто використовуваних платформ. Він поширюється за безкоштовною ліцензією Python Software Foundation, що дозволяє використовувати його без обмежень у будь-яких додатках, включаючи власні.

Python найчастіше порівнюють з Ruby та Perl. Ці мови мають приблизно однакову швидкість виконання програми і також інтерпретуються. Як і Ruby, Python є добре продуманою структурою ООП.

Порівняно з Perl, Python також може успішно використовуватися для написання різних сценаріїв. У комерційних програмах швидкість виконання програм Python часто порівнюють із швидкістю виконання програм на мові програмування Java.

Багата стандартна бібліотека - одна з цікавих речей у Python. Існують інструменти для роботи з багатьма форматами та мережевими протоколами Інтернету, наприклад, модулі для написання HTTP-клієнтів та серверів, для роботи з XML, для створення та аналізу поштових повідомлень тощо. Є модулі для роботи з кодуваннями тексту, звичайні вирази, серіалізація даних, архіви, криптографічні протоколи, мультимедійні формати, є також підтримка модульного тестування тощо. Набір модулів для роботи з операційною системою дозволяє писати кросплатформні програми. Поряд зі стандартною бібліотекою існує велика кількість сторонніх бібліотек, які значно скорочують час розробки та збільшують функціональність. Python і переважна більшість бібліотек для нього постачаються у вихідних кодах і є безкоштовними. Більше того, на відміну від багатьох відкритих систем, ліцензія не накладає жодних зобов'язань, крім авторських прав, і не обмежує використання Python у комерційній розробці.

Серед недоліків - низька продуктивність (хоча багато програм та бібліотек, реалізованих у Python для інтеграції з іншими мовами програмування, надають можливість використовувати іншу мову для написання критичних розділів, отже, для підтримки сумісності з багатьма мовами програмування модуль обробки знімків є реалізовано в C ++), глобальне блокування інтерпретатора та неможливість модифікації вбудованих класів. Ці недоліки не є серйозними для розробки та функціонування ПМ ВІЗ.

Perl

Perl - це загальна, динамічна, інтерпретована мова програмування високого рівня, яка спочатку була призначена для маніпулювання текстом, але зараз активно використовується для широкого кола завдань, включаючи веб-розробку, системне адміністрування, мережеве програмування, біоінформатику, ігри та розробка графічного інтерфейсу користувача.

Основними перевагами мови є підтримка різних парадигм (процедурний, функціональний та об'єктно-орієнтований стилі програмування), вбудована підтримка обробки тексту, велика колекція сторонніх модулів та управління пам'яттю (відсутність збирача сміття на основі циклу).

Perl успадкував багато властивостей від мов сценаріїв оболонки C, AWK та UNIX. Головною особливістю мови вважають її багаті можливості для роботи з текстом, включаючи роботу із вбудованими регулярними висловами синтаксису. Perl має безліч вбудованих функцій, які надають набір інструментів, що часто використовуються для програмування оболонки, таких як виклик системних служб або сортування.

Мови опису схем. XML

Поняття про DOM.

DOM (Document Object Model) - об'єктна модель документа. Це незалежний від платформи і мови програмний інтерфейс, що дозволяє програмам отримувати доступ до вмісту документів, а також змінювати вміст, структуру і вид документів.

В рамках DOM будь-який документ представляється у вигляді дерева вузлів. Кожен вузол являє собою елемент, атрибут, текстовий, графічний або будь-який інший об'єкт. Вузли між собою перебувають у відношенні «батько-нащадок».

Спочатку різні браузери мали власні моделі DOM, не сумісні з іншими. Для того, щоб забезпечити взаємну і зворотну сумісність, консорціум W3C класифікував цю модель по рівнях, для кожного з яких була створена своя специфікація. Всі ці специфікації об'єднані в загальну групу, що носить назву W3C DOM.

Об’єктна модель XML-документа

XML служить метамовою для опису структури інших мов. Взаємозв'язок між SGML, XML, HTML і деякими іншими мовами показана на наступній діаграмі

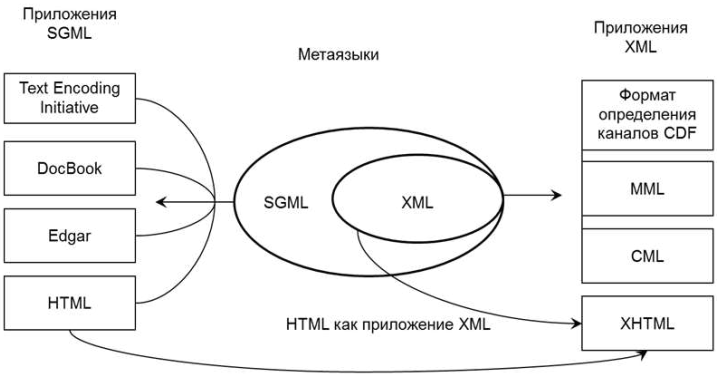


Рисунок 3.2.3 - Об’єктна модель XML-документа

Документ XML вважається правильно побудованим, якщо він відповідає всім синтаксичним правилам XML.

Перевірка дійсності документа передбачає виконання таких дій:

- Перевірка використання тільки заданого набору дескрипторів.
- Перевірка повної відповідності порядку проходження елементів і атрибутів змісту документа або певними правилами.
- Контроль типів даних (досягається при використанні відповідної схеми).

- Контроль цілісності даних для забезпечення оптимального обміну інформацією через Веб з допомогою транзакцій.
- Розглянемо тепер основні синтаксичні правила побудови XML документів.
- XML документ містить один і тільки один кореневий елемент, що містить всі інші елементи
- Дочірні елементи, що містяться в кореневому елементі, повинні бути правильно вкладені.
- Імена елементів підкоряються правилам:
- Ім'я починається з букви, знака підкреслення або двокрапки.
- Після першого символу в імені можуть бути букви, цифри, знаки переносу, підкреслення, крапка або двокрапка.
- Імена не можуть починатися з буквосполучення XML.

Описані вище правила дозволяють контролювати тільки *формальну правильність XML документа*, але не змістовну. Для вирішення другого завдання використовуються так звані *схеми.*

XML (eXtensible Markup Language) - рекомендований W3C, як мову розмітки. XML, це текстовий формат, призначений для зберігання структурованих даних, для обміну інформацією між програмами, а також для створення на його основі спеціалізованих мов розмітки.

Мова XML як і люба мова, має синтаксичні правила і стандарти.

XML має такі переваги:

- Це людино-орієнтований формат документа, він зрозумілий як людині, так і комп'ютера.
- Підтримує Юнікод.
- У форматі XML можуть бути описани основні структури даних - таки е як записи, списки і дерева.
- Це самодокументуємий формат, який описує структуру і імена та значення полів.
- Має строго певний синтаксис і вимоги до аналізу, що дозволяє йому залишатися простим, ефективним і несуперечливим.

Широко використовується для зберігання і обробки документів;

Це формат, заснований на міжнародних стандартах;

- Ієрархічна структура XML підходить для опису практично будь-яких типів документів;
- Являє собою простий текст, вільний від ліцензування і будь-яких обмежень;
- Не залежить від платформи;
- Є підмножиною SGML, для якого накопичений великий досвід роботи і створені спеціалізовані додатки;

На жаль, правила створення структури документу дозволяють контролювати тільки формальну частину XML документа, але не змістовну. Для вирішення другого завдання використовуються так звані схеми**.**

XML-Схеми

Схема чітко визначає ім'я та структуру кореневого елемента, включаючи специфікацію всіх його дочірніх елементів. Програміст може задати, які елементи і в якій кількості обов'язкові, а які - необов'язкові. Схема також визначає, які елементи містять атрибути, допустимі значення цих атрибутів, в т.ч. значення за замовчуванням.

Найчастіше для опису схеми використовуються наступні специфікації:

- DTD (Document Type Definition) - мова визначення типу документів.
- XDR (XML Data Reduced) - діалект XML, розроблений Microsoft.
- XSD (мова визначення схем XML) - рекомендована консорціумом W3C.

XML документ відрізняється від HTML документа також і тим, як він відображається у веб-браузері. Без використання CSS або XSL XML-документ відображається як простий текст в більшості веб-браузерів. Деякі веб браузери, такі як Internet Explorer, Mozilla і Firefox відображають

структуру документа у вигляді дерева, дозволяючи згортати і розгортати вузли за допомогою натискань клавіші миші.

Найбільш поширені три способи перетворення XML-документа в відображуваний користувачеві вигляд:

- Застосування стилів CSS.
- Застосування перетворення XSLT.
- Написання на якомусь мові программиро вання обробника XML-документа.

Головною задачею XML- схем є те, що реалізуя перевірки даних в додатках, які генеруються приймаючі транзакції на рівні документа, можна оптимізувати виконання для забезпечення максимальної швидкодії. Також можна перевірити відповідність полів і правильність записів на рівні примірників XML.

Для перевірки дійсності XML документа можна використовувати спеціальні валідатори, наприклад W3C валідатор (http://validator.w3.org/).

Для перевірки схем також існують спеціальні валідатори, наприклад XML Schema валідатор (http://www.w3.org/2001/03/webdata/xsv).

Згідно специфікації W3C XML програма повинна припинити обробку XML документа, як тільки буде виявлена помилка в цьому документі.

Інтерфейси взаємодії веб- застосувань .Технологія AJAX

Програмні продукти прикладного характеру, які орієнтовані на кінцевого користувача, працюють у діалоговому режимі взаємодії з користувачем. В цьому режимі під впливом користувача здійснюються запуск функцій (методів) обробки, зміна властивостей об'єктів, проводиться налаштування параметрів видачі інформації на друк.

Системи, що підтримують діалогові процеси, класифікуються на:

- системи з жорстким сценарієм діалогу
- стандартизоване представлення інформації обміну;

- дескрипторні системи
- формат ключових слів повідомлень;
- тезаурусні системи - семантична мережа дескрипторів, що утворюють словник системи (аналог - гіпертекстові системи);
- системи з мовою ділової прози - подання повідомлень на мові, природному ;
- для професійного користування.

Найбільш прості для реалізації та поширені діалогові системи з жорстким сценарієм діалогу.

Опис сценарію діалогу: блок-схема, в якій передбачені блоки видачі повідомлень і обробки отриманих відповідей; орієнтований граф, вершини якого - повідомлення і виконувані дії, дуги - зв'язок повідомлень; словесний опис; спеціалізовані об'єктно-орієнтовані мови побудови сценаріїв.

Для створення діалогових процесів і інтерфейсу кінцевого користувача найбільш підходять об'єктно-орієнтовані інструментальні засоби розробки програм.

У складі інструментальних засобів СУБД містяться побудовники меню, з допомогою яких створюється орієнтована на кінцевого користувача сукупність режимів і команд в вигляді головного меню і вкладених підменю. Конструктор екранних форм СУБД використовується для розробки форматів екранного введення і редагування даних бази даних і вхідної інформації, керуючої роботою програмного продукту.

У ряді СУБД і електронних таблиць, текстових редакторів існують різні типи діалогових вікон, містять різноманітні об'єкти управління: тексти повідомлення; поля введення інформації користувача; списки можливих альтернатив для вибору; кнопки і т.п.

У середовищі електронних таблиць і текстових редакторів є можливості налаштування головних меню (видалення непотрібних,

додавання нових режимів і команд), створення системи підказок за допомогою вбудованих засобів і мов програмування.

Сьогодні більшість інформаційних систем в тій чи іншій мірі використовують бази даних. До почав 90-х років існувало кілька різних постачальників баз даних, кожен з яких мав власний інтерфейс. Якщо додатком було необхідно обмінюватися даними з декількома джерелами даних, то для взаємодії з кожною з баз даних, було необхідно написати окремий код. З метою вирішення цієї проблеми Майкрософт і ряд інших компаній створили стандартний інтерфейс для отримання і відправки даних джерел даних різних типів. Цей інтерфейс отримав назву open database connectivity (ODBC).

Види веб-інтерфейсів користувача.

Перед розробниками веб-інтерфейсів в будь-якому проекті поставлено завдання створення саме дружнього по відношенню до користувача інтерфейсу. Однак це не завжди таке просте завдання, як може здатися на перший погляд, і часом вимагає не малого досвіду проектування. Головні вимоги це: - зручність, практичність і інтуїтивна зрозумілість. Саме в цей момент вступають в гру такі поняття як UX і UI дизайн.

Технології побудови призначених для користувача інтерфейсів на базі шаблонів, реалізованих на мовах розмітки, почали повсюдно застосовуватися з середини 1990-х. Основні переваги шаблонів - гнучкість і широта можливостей створення динамічних призначених для користувача веб-інтерфейсів, особливо з точки зору розробки структури і планування. Спочатку в таких інструментарій використовувалися шаблони, в яких планування і структура UI задавалися за допомогою мови розмітки, а прив'язка до даних здійснювалася за допомогою невеликих блоків на мові високого рівня (Java, C #, PHP, Python і т. Д.). Останні могли використовуватися в комбінації з розміткою; наприклад, шляхом впровадження тегів розмітки в цикл на Java могли створюватися ітеративні

візуальні елементи на зразок таблиць і списків. Необхідність частої зміни синтаксису всередині веб-сторінки ускладнювала розробку і корекцію коду для програмістів, тому близько десяти років тому почався перехід з мов високого рівня на спеціалізовані бібліотеки тегів розмітки і мови виразів, створені для конкретних веб-технологій.

Теги розмітки стали використовувати для реалізації типових функцій веб-додатків, а вираження - для доступу до даних і доступу до більшості функцій, які зберігаються в серверних об'єктах. Типовий представник цієї групи - технологія JavaServer Pages (JSP), бібліотека тегів якої JSP Standard Tag Library підтримує такі завдання, як: маніпуляція з XML-документами, цикли, умови, опитування СУБД (прив'язка до даних) і інтернаціоналізація (форматування даних). Мова виразів JSP - EL, службовець засобом прив'язки до даних, пропонує зручну нотацію для роботи з об'єктами і властивостями додатки.

Значна частка інструментаріїв для створення UI базується на об'єктно-орієнтованої моделі. Зазвичай ці інструментарії пропонують бібліотеку готових елементів UI, і їх головними перевагами є простота складання багаторазово використовуваних блоків з простих компонентів і інтуітивно зрозумілий, гнучкий процес програмування поведінки та взаємодії, заснований на обробниках подій. У цих інструментаріях всі завдання розробки UI вирішуються з використанням спеціалізованих об'єктних API. До даної категорії відносяться середовища: Visual Basic, MFC, AWT, Swing, SWT, Delphi, Google Web Toolkit, Cocoa Touch UIKit, Vaadin і ін.

Гібридні технології відносно нові в світі розробки UI загального призначення - поряд з шаблонами і мовами виразів в подібних інструментарій застосовується об'єктний API. Типовий представник - JavaServer Faces: бібліотеки тегів служать для опису структури і планування, а також для форматування даних; мова виразів - для прив'язки

елементів і подій до серверних об'єктів і коду додатків; об'єктний API - для відображення елементів, управління їх станом, обробки подій і контролю введення. Інші популярні інструментарії в цій категорії: ASP.NET MVC, Apache Wicket, Apache Tapestry, Apache Click і ZK Framework.

Технологія AJAX

AJAX (Asynchronous JavaScript And XML) — підхід до побудови користувацьких інтерфейсів вебзастосунків, за яких вебсторінка, не перезавантажуючись у фоновому режимі, надсилає запити на сервер і сам звідти довантажує потрібні користувачу дані.

AJAX - це концепція використання декількох суміжних технологій, орієнтірована на розробку високоінтерактівних додатків , які швидко реагують на дії користувача, що виконують велику частину роботи на стороні клієнта і взаємодіючих з сервером за допомогою позасмугових звернень. Позасмуговими зверненнями називається запит до сервера, який призводить до оперативного оновлення сторінки замість її заміни. Позасмуговий виклик HTTP - це HTTP запит, який видається за межами вбудованого модуля, що забезпечує відправку форм HTTP.

AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX-підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

- Використання DHTML для динамічної зміни змісту сторінки.
- Використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю
- альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT>

Використання цих підходів дозволяє створювати набагато зручніші вебінтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. AJAX — асинхронний, тому користувач може

переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з'єднання з сервером). Використання AJAX стало популярним після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google Maps і Google Suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

1. Класична модель підходу до вебзастосунку:

- Користувач заходить на вебсторінку і натискає на який-небудь її елемент;
- Браузер надсилає запит серверу;
- У відповідь сервер генерує повністю нову вебсторінку і відправляє її браузеру і т. д.
- З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

2. Модель AJAX:

- Користувач заходить на вебсторінку і натискає на який-небудь її елемент.
- Браузер відправляє відповідний запит на сервер.
- Сервер віддає тільки ту частину документа, яка змінилася.

Позитивні сторони такого підходу:

- економія коштів – оновлю ються окремі дані, а не вся сторінка; є те, що її використання дає змогу не лише переглядати веб-ресурси мережі Інтернет, а й завантажувати власні, здійснювати обмін цими ресурсами з іншими користувачами
- економія часу – швидкість завантаження сторінки значно збільшується, оскільки завантажується лише та частина сторінки, яка була змінена;

• зменшення навантаження на сервер – використання мови програмування JavaScript дає змогу значно зменшити навантаження на сервер, оскільки виконання сценаріїв, написаних мовою, відбувається у веб-оглядачі.

Звісно, існують і недоліки використання AJAX, зокрема деякі проблеми пов'язані з функціонуванням скриптів, створених мовою JavaScript, і стандартних функцій веб-оглядача, наприклад:

– динамічно створені сторінки не зберігаються в історію відвідувань веб-оглядача, тому неможливо повернутись і переглянути ті сторінки, які користувач відвідав раніше. Але цю проблему частково можна вирішити, використовуючи додаткові скрипти;

– динамічно створені сторінки неможливо зберегти в закладинки веб-оглядача, оскільки вони не мають URL-адреси. Частково вирішити цю проблему можна за допомогою динамічної зміни ідентифікатора фрагмента (частини URL-адреси після знака «#»), що підтримується багатьма веб-оглядачами.

Існує також проблема з тим, що динамічно створені сторінки не можуть бути проіндексовані пошуковими системами, а тому знайти таку сторінку неможливо. Цю проблему можна вирішити шляхом пошуку та створення альтернативних способів доступу до змісту сайта.

Проблеми не гальмують використання підходу AJAX як технологічної основи Веб 2.0, про що свідчить їх популярність серед користувачів мережі Інтернет. Швидке отримання інформаційних даних поряд з невеликими затратами коштів відкриває великі можливості використання технологій Веб 2.0 як в повсякденному житті, так і в навчанні.

3.3. ІНТЕГРАЦІЯ ТА ВЗАЄМОДІЯ У ВЕБ-МЕРЕЖІ

3.3.1. Архітектура веб-додатків ASP. NET, JSP

ASP.NET є платформою для створення веб-додатків і веб-сервісів, що працюють під управлінням IIS.

ASP.NET (Active Server Pages .NET) — це безкоштовна вебплатформа, створена фахівцями Microsoft для проєктування інтерактивних вебзастосунків, які працюють на платформі .NET.

За допомогою ASP.NET створюють різні вебпрограми та сервіси, включаючи:

- Вебсайти. Прості та статичні сайти, які не потребують складної логіки та обробки даних.
- Вебпрограми. Складні програми з високим ступенем логіки та обробки даних, які можуть бути використані для управління бізнес-процесами, продажем товарів чи наданням послуг.
- Сервіси. Програми, які можуть бути використані для обміну даними між різними програмами та системами.
- API. За допомогою ASP.NET можна створювати API для обміну даними з іншими програмами та службами.
- Мобільні застосунки. ASP.NET також може бути в бути використаний для створення мобільних програм з використанням фреймворку Xamarin. Цей фреймворк дозволяє розробляти кросплатформні нативні програми для iOS та Android.

Компанія Майкрософт ASP.NET побудувала на базі CLR (Common Language Runtime), який є основою всіх додатків .NET. Розробники можуть створювати код для ASP.NET, використовуючи мови програмування: C #, Visual Basic.NET, JScript.NET та інші.

У файлах ASP.NET включається код на таких мовах програмування як C #, JScript.NET, VisualBasic.NET, що дозволяє застосовувати безпосередньо у веб-додатків можливості об'єктно-орієнтованого

програмування. Також значно скорочується обсяг коду, написаного вручну за рахунок застосування серверних об'єктів , автоматично генеруючих код елементів управління HTML. Можливо використання стандартною середовища розробки *Visual Studio. NET,* тобто ASP.NET має перевагу в швидкості в порівнянні зі сценарними технологіями, так як при першому зверненні код компілюється і поміщається в спеціальний кеш, а згодом тільки виконується, не вимагаючи витрат часу на парсинг, оптимізацію, і т.

Файли *ASP. NET* обробляються бібліотекою *aspnet _ isapi. Dll* (а не *asp. Dll),* яка, в свою чергу, використовує для виконання коду *технологію. NET.*

Бібліотека базових класів. NET містить простору імен 3 основних груп:

- елементи *web-додатків* (протоколи, безпека та ін.);
- елементи *графічного інтерфейсу* (WebForms);
- *web- служби.*

ASP.NET використовує можливості стандартного середовища розробки *Visual Studio.Net,* і зокрема клас и бібліотеки FCL (Framework Class Library).

В основу розробки веб-додатків на ASP.NET покладено модель *поділу коду уявлення і коду реалізації,* рекомендована Майкрософт при створенні динамічних документів за допомогою програмних кодів. Це робиться шляхом розміщення програмного коду або в окремий файл, або всередині спеціального тега сценаріїв.

Файл такого роду зазвичай має розширення *. *Aspx. Cs* (*. Aspx. Vb) і має ім'я, що збігається з ім'ям основного ASPX файлу. В принципі такий підхід дозволяє веб-дизайнеру сконцентруватися роботі з кодом розмітки документа з мінімальними змінами програмного коду, в звичайному ASP впроваджуваного безпосередньо в код розмітки.

Розробнику веб-додатків на ASP.NET доступні класи, що входять в наступні простору імен:

Простір імен	Зміст
System.Web	Організація взаємодії web-клієнта (браузера) з web-сервером (запит-відповідь, cookie і та ін.)
System.Web.Caching	Підтримка кешування при роботі web-додатків
System.Web.Configuration	Налаштування web-додатки відповідно до файлами конфігурації проекту
System.Web.Security	Реалізація системи безпеки web-додатків
System.Web.Services	Организ ація роботи web-сервісів
System.Web.Services.Description	
System.Web.Services.Discovery	
System.Web.Services.Protocols	
System.Web.UI	Побудова графічного інтерфейсу користувачів web-додатків
System.Web.UI.WebControls	
System.Web.HtmlControls	

Взаємодія користувача з веб-додатком, реалізованому на ASP.NET включає в себе такі процеси:

- При запиті сторінки *ASPX* ініціюється подія *Page_Init,* що виробляє початкову ініціалізацію сторінки та її об'єкта.
- Далі ініціюється подія *Page_Load,* яке може бути використано, наприклад для установки початкових значень для елементів управління. При цьому також можна визначити чи була завантажена сторінка вперше або звернення до неї здійснюється повторно в рамках зворотного відсилання у відповідь на події, пов'язані з елементами управ ління, розміщеними на сторінці; т. е. перевірити властивість *Page.IsPostBack.*

- Далі виконується перевірка валідності елементів сторінки з погляду коректності введених користувачем даних.
- І, нарешті, обробка всіх подій, пов'язаних з діями користувача з моменту останнього зворотнього відсилання.

Для збереження даних веб-сторінки в проміжках між зверненнями до неї в ASP.NET використовують стан відображення (view state).

Якщо дані, введені в веб-форму, необхідно зробити доступними іншим веб-формам того ж додатка, ці дані необхідно зберегти в об'єктах *Application* і *Session.* Об'єкти *Application* доступні всім користувачам програми та можуть розглядатися як глобальні змінні, звернення до яких можливо з будь-яких сеансів. Об'єкти *Session* доступні тільки в рамках одного сеансу, і тому вони виявляються доступними тільки одному користувачеві.

Принцип роботи ASP.NET

В основі роботи ASP.NET лежить принцип клієнт-серверної архітектури та протокол HTTP:

1. спочатку йде запит від клієнта;
2. коли користувач надсилає запит на вебсервер, ASP.NET приймає запит та починає його обробку;
3. проходить ланцюжок обробників, які можуть виконувати різні завдання, наприклад, перевірку авторизації або обробку даних форми;
4. далі ASP.NET виконує код на сервері, який може генерувати HTML, CSS, JavaScript та інші дані для надсилання назад клієнту;
5. після генерації сторінки ASP.NET відправляє її назад клієнту у вигляді відповіді на його запит.

Далі платформа підтримує стан між запитами та відповідями, такі як використання куків (cookies) чи сесій (sessions). Стан може

використовуватися, наприклад, для збереження інформації про користувача**,** щоб уникнути повторної автентифікації кожного запиту.

Якщо під час обробки запиту виникають помилки, ASP.NET може створювати спеціальну сторінку помилок. Ця сторінка дозволяє розробникам розібратися із ситуацією: зрозуміти, що сталося та як виправити помилку.

Моделі розробки в ASP.NET визначають, як буде організовано застосунок. Існує багато різних моделей розробки, які можуть бути використані в ASP.NET в залежності від задачі, яку потрібно вирішити. Розглянемо коротко види моделей в ASP.NET:

1. Model-View-Controller

Одна з найпоширеніших моделей розробки ASP.NET.

Model-View-Controller (MVC) — це популярний шаблон проєктування, який розділяє застосунок на три основні компоненти**:** (1) модель, (2) представлення та (3) контроллер**:**

- модель містить дані та логіку програми, яка керує цими даними;
- представлення визначає, як дані відображатимуться на екрані;
- а контроллер управляє потоком даних між моделлю та представленням (рис 3.3.1).

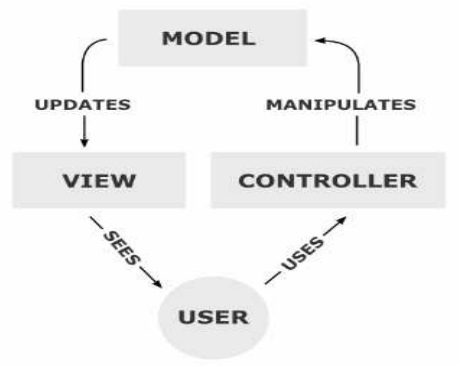


Рисунок 3.3.1 -Model-View-Controller

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки самої моделі від її візуалізації тобто виду, уявлення. Це дає можливість підвищити можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах.

MVC призначена для розробки складних вебзастосунків з високим ступенем логіки та обробки даних. Також модель часто обирають для створення мобільних та односторінкових проєктів.

Для реалізації схеми «Model-View-Controller» використовується досить велика кількість шаблонів проєктування, основні з яких - «спостерігач», «стратегія», «компоновщик».

Іноді дії треба виконати не над однією моделлю, а над декількома одночасно. Наприклад, при видаленні користувача треба видалити всі його статті та коментарі. В результаті доводиться створювати контролер, який описує операції не тільки над моделлю до якої він відноситься (у прикладі - користувачі), але над моделями до яких безпосереднього відношення він не має. Таким чином з'являються не явні залежності моделі MVC (рис.3.3.2).

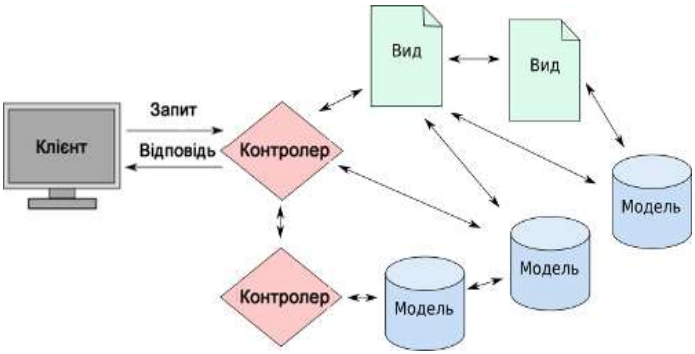


Рис.3.3.2 Не явні залежності моделі MVC

Движок працює по HMVC моделі, де кожна тріада MVC являє собою незалежний компонент, який може включати в себе не тільки Model-View-Controller, а так само всі інші складові сайту (config, init, css, js, image ...). Model це скоріше Бібліотека, яку може використовувати компонент. Моделі будуються не на структурі компонентів, а на структурі БД. Прикладами компонентів є: контакти, посторінковий бар, пошук, меню, слайдер та ін. А прикладами моделей є: модель для роботи з користувачами, модель для роботи з новинами, модель для роботи з товарами та ін. Концептуальна модель роботи системи представлена на рисунку 3.3.3.

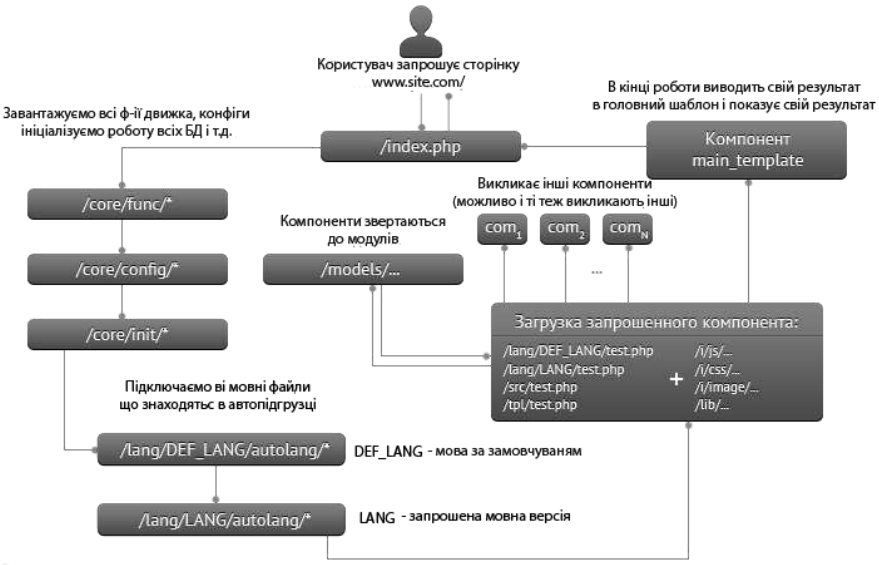


Рис. 3.3.3 - Концептуальна модель роботи системи

Не дивлячись на певну складність схеми, ідейно все дуже просто: виклик index.php, він включає всі файли в core, підключає мовні файли → src → tpl → виводить результат в main_template

2. Web Forms

Поширена модель розробки ASP.NET, але вона використовується в основному на старих проєктах.

У Web Forms програма розробляється як набір вебсторінок, кожна з яких містить розмітку HTML та код мовою C# чи VB.NET.

Вебсторінки можуть бути пов'язані між собою за допомогою елементів керування, таких як кнопки чи посилання.

Загалом існує тенденція відмови від Web Forms на користь ASP.NET MVC, яка є більш просунутою моделлю та надає найкращі можливості для тестування та керування HTML-розміткою.

3. Web API

Web API дозволяє створювати RESTful-вебсервіси, які будуть обробляти HTTP-запити та повертати HTTP-відповіді. За допомогою Web API можна виконувати форматування відповідей (в JSON, XML та інших форматах), конфігурувати маршрутизацію запитів, призначати фільтри дій та багато іншого.

За допомогою інструмента Web API можна створювати різноманітні вебсервіси, наприклад, сервіси авторизації або послуги з обробки платежів, сервіси керування вмістом та багато іншого. Він також може бути використаний для створення мобільних програм та інших клієнтських програм, які отримують дані з вебсервісів.

Серверні елементи управління ASP.NET

Важливою особливістю ASP.NET є використання *серверних елементів управління* на веб-сторінці (елементи *WebForm),* які є фактично тегами,

зрозумілими веб-сервера. Ці елементи визначені в просторі імен *System. Web. UI. WebControls.*

Прийнято виділяти три типи серверних елементів управління:

- Серверні *елементи управління HTML* - звичайні HTML теги.
- *Елементи управління веб-сервера* - нові теги ASP.NET.
- Серверні *елементи управління для перевірки даних (валідації)* - застосовуються для валідації вхідних даних від клієнтського застосування (зазвичай веб-браузера).

Переваги від використання таких елементів при розробці веб-додатків:

- Скорочується кількість коду, написаного вручну (що особливо помітно в складних елементів документа). Елемент просто «перетягується» з панелі інструментів, після чого виконується налаштування його параметрів у спеціальному вікні. При цьому всі зміни автоматично заносяться безпосередньо в *. *Aspx* файл.
- З програмної точки зору кожного з цих елементів управління відповідає певний клас в бібліотеці базових класів. NET, що дозволяє писати для них такий же код як і для будь-яких інших класів.
- Для будь-якого елементу управління WebForm визначений набір подій, оброблюваних на веб-сервері.
- Для будь-якого елементу управління WebForm надається можливість для перевірки введення даних користувачем.

За замовчуванням серверні *елементи управління HTML* в ASP.NET файлах розглядаються як текст. Для їх програмування потрібно додавання атрибуту *runat = "server"* у відповідний HTML елемент. Крім того, всі серверні елементи управління HTML повинні бути розміщені всередині області дії тега *<form>*, також має атрибут *runat = "server"*.

Подібно серверним елементам управління HTML *елементи управління веб-*

сервера також створюються на вебсервері і припускають додавання атрибу

ту *runat = "server"*. Однак вони можуть і не відповідати конкретним елементам HTML, але представляти більш складні елементи.

Загальний синтаксис для опису таких елементів:

<Asp: *min_елемента* id = *"ідентифікатор"* runat = "server" />

Серверні елементи валідації застосовуються для перевірки введених користувачем даних.

Мають наступний синтаксис:

<Asp: *min_елемента* id = *"ідентифікатор"* runat = "server" />

Робота з джерелами даних в ASP. NET

В ASP. NET використовуються два елементи управління WebForm для управління відображенням даних, одержуваних з джерела даних:

- *DataGrid* - елемент управління, що відображає вміст об'єкта ADO. NET *DataSet* у вигляді таблиці.
- *DataList* - елемент управління для вибору значень, що заповнюються з джерела даних.

Якщо необхідно відобразити дані, отримані за запитом користувача з джерела даних, у вигляді таблиці на веб-сторінці, то ASP. NET надає в розпорядження веб-програміста зручний елемент управління *DataGrid.*

Технологія Java Servlet

Сервлети Java - це програмні модулі Java на стороні сервера, які обробляють і відповідають на запити клієнтів та реалізують інтерфейс сервлета. Це допомагає покращити функціональність веб-сервера з мінімальними накладними витратами, обслуговуванням та підтримкою.

Сервлет виступає посередником між клієнтом і сервером. Оскільки сервлетні модулі працюють на сервері, вони можуть приймати та реагувати на запити клієнта. Об'єкти запиту та відповіді сервлету пропонують зручний спосіб обробляти HTTP-запити та надсилати текстові дані назад клієнту.

Оскільки сервлет інтегрований з мовою Java, він також має всі функції Java, такі як висока портативність, незалежність від платформи, безпека та підключення до бази даних.

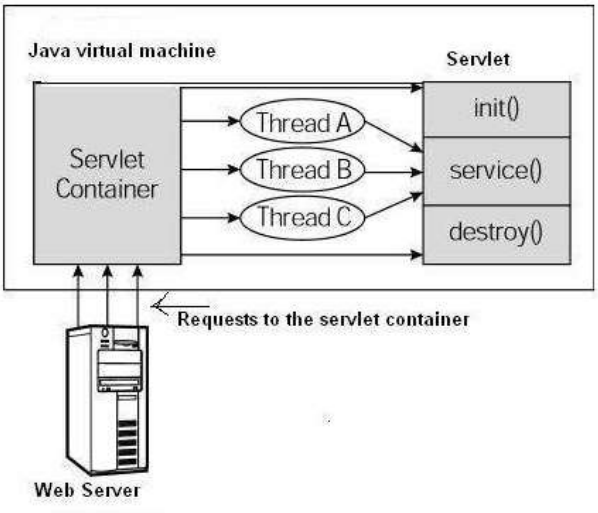


Рисунок 3.3.4. Життєвий цикл сервлету

Життєвий цикл сервлета (рис.3.3.4) можна визначити як весь процес від його створення до знищення. Нижче наведені шляхи, за якими рухається сервлет:

- Сервлет ініціалізується викликом методу init().
- Сервлет викликає метод service() для обробки запиту клієнта.
- Сервлет завершується викликом методу destroy().

- Нарешті, сервлет - це сміття, зібране збирачем сміття JVM.

А тепер давайте детально обговоримо методи життєвого циклу.

Метод init викликається лише один раз. Він викликається лише тоді, коли створюється сервлет, а потім не викликається для будь-яких запитів користувачів. Отже, він використовується для одноразової ініціалізації, як і в методі init аплетів. Зазвичай сервлет створюється, коли користувач вперше викликає URL-адресу, що відповідає сервлету, але ви також можете вказати, що сервлет завантажується при першому запуску сервера.

Коли користувач викликає сервлет, створюється одиничний екземпляр кожного сервлету, при цьому кожен запит користувача призводить до нового потоку, який передається до doGet або doPost відповідно. Метод init() просто створює або завантажує деякі дані, які будуть використовуватися протягом усього життя сервлета.

Метод service() є основним методом для виконання фактичного завдання. Контейнер сервлету (тобто веб-сервер) викликає метод service() для обробки запитів, що надходять від клієнта (браузери), і для запису відформатованої відповіді назад на клієнта. Щоразу, коли сервер отримує запит на сервлет, сервер створює новий потік і викликає службу. Метод service() перевіряє тип запиту HTTP (GET, POST, PUT, DELETE тощо) і викликає методи doGet, doPost, doPut, doDelete тощо. Метод service() викликається контейнером, а метод service викликає методи doGet, doPost, doPut, doDelete тощо. Отже, ви не маєте нічого спільного з методом service (), але замінюєте doGet () або doPost () залежно від того, який тип запиту ви отримуєте від клієнта.

DoGet () та doPost () є найбільш часто використовуваними методами у кожному запиті на обслуговування. Ось підпис цих двох

методів. Запит GET є результатом звичайного запиту на URL-адресу або форми HTML, у якій не вказано METHOD, і його слід обробляти методом doGet (). Запит POST є результатом HTML-форми, яка конкретно перелічує POST як МЕТОД, і його слід обробляти методом doPost ().

Метод destroy() викликається лише один раз в кінці життєвого циклу сервлета. Цей метод дає сервлету можливість закрити підключення до бази даних, зупинити фонові потоки, записати на диск списки файлів cookie або кількість звернень та виконати інші подібні дії з очищення. Після виклику методу destroy() об'єкт сервлета позначений для збору сміття.

На рисунку 3.3.4. зображено типовий сценарій життєвого циклу сервлета. Спочатку HTTP-запити, що надходять на сервер, делегуються контейнеру сервлетів. Контейнер сервлета завантажує сервлет перед викликом методу service(). Потім контейнер сервлета обробляє кілька запитів, створюючи кілька потоків, кожен потік виконує метод service ()

Для більшої інтерактивності і продуктивності розроблений підхід до розробки веб -додатків AJAX. При використанні Ajax, сторінки веб-додатку здатні відправляти веб-запити до сервера у фоновому режимі і не потребують довантаження необхідних даних з сервера. Це значно пришвидшує роботу.

Для створення вебзастосунків використовуються різноманітні серверні технології та мови програмування(таблиця 3.3.1).

Таблиця 3.3.1 Таблиця серверних технології та мов програмування

Назва	Ліцензія	Вебсервер
ASP	власницька	спеціалізований

Назва	Ліцензія	Вебсервер
ASP.NET	власницька	спеціалізований
Java	вільна	безліч, зокрема вільних
Groovy	вільна	практично будь-який
Perl	вільна	практично будь-який
PHP	вільна	практично будь-
Python	вільна	практично будь-який
Ruby	вільна	практично будь-який

Технологія JSP (Java Server Pages) − це технологівя створення серверних сторінок Java. Специфікація JSP є розширенням Java Servlet API для генерації динамічних веб-сторінок на вебсервері. Така крос-платформа є альтернативою технології ASP корпорації Microsoft. Специфікація Sun за назвою JSF (Java Server Faces) реалізує технологію JSP, що описує правила створення вебдодатків зі зручним для користувача інтерфейсом та орієнтована на розробку серверних компонентів створення інтерфейсу.

Однією з перших технологій створення веб-застосувань, які виконуються сервером, була Common Gateway Interface (CGI) технологія . Вона дозволила розробку і виконання серверних застосувань, звернення до яких відбувається за допомогою зазначеного в URL імені (та параметрів). Залежно від обраного протоколу вхідною інформацією таких веб-додатків вважають безпосередньо код HTTP-заголовка або запит пошукової системи. CGI-застосування − це консольні додатки, які генерують HTML-код, переданий браузеру. Серед інших популярних технологій, які реалізують створення веб- сторінок із фрагментами коду, виконуваного на сервері, виділимо некомерційну, вільно розповсюджувану технологію PHP (Personal Home Pages). Ця технологія заснована на використанні CGI-

застосувань, що інтерпретують впроваджений у HTML-сторінку код на скриптовій мові.

3.3.2. Розробка веб-контенту засобами CMS/CMF

CMS (content management system) – система динамічного відновлення й редагування змісту Інтернет-сайту.

Система управління контентом (Content management system, CMS) - комп'ютерна програма, яка використовується для створення, редагування, управління та публікації контенту деяким систематичним чином.

Система управління веб-контентом дозволяє користувачам керувати цифровими компонентами веб-сайту без попереднього знання мов розмітки або веб-програмування. WCMS забезпечує інструменти для співпраці, створення та адміністрування, що допомагає керувати цифровим вмістом. На відміну від інших систем управління контентом, які займаються вмістом, призначеним як для Інтернету, так і для друку, WCMS обробляє виключно веб-вміст.

Зазвичай такі системи використовуються для зберігання і публікації великої кількості документів, зображень, музики або відео. Приклади CMS представлені на рисунку 3.3.5.

Рисунок 3.3.5.- Приклади CMS

Система управління веб-контентом (Web content management system, WCMS або Web CMS) - програмне забезпечення *CMS* класу, реалізоване зазвичай у вигляді веб-додатки, і призначене для створення, і управління HTML вмістом.

WCMS зазвичай використовується для управління і контролю великими, динамічно змінюваними колекціями веб-матеріалу*(HTML* документами і пов'язаними з ними картинками). Така система спрощує процес створення, управління, редагування контенту і багато інших важливі завдання, пов'язані з підтримкою цих процесів.

WCMS надає наступні можливості:

- *Застосування автоматичних шаблонів відображення* (в *HTML* або *XML* форматі), автоматично застосовуються до нового або існуючого контенту. Тим самим вигляд всіх документів може здаватися з одного місця.
- *Простота редагування контенту.* Користувачеві досить легко створювати і управляти контентом, оскільки йому або взагалі не потрібно знання мов програмування або мов розмітки, або потрібно мінімальне знання таких.
- *Масштабованість.* Можливість розширення функціональності існуючого сайту шляхом установки поставляються з дистрибутивом *WCMS плагінів* і *модулів.*
- *Управління документами.* Є засоби управління життєвим циклом документів з моменту створення до вилучення.
- *Візуалізація контенту.* Будь-який користувач може працювати з віртуальною копією всього веб-сайту, безлічі документів або кодами програм, що дозволяє *побачити* всі*зміни* безлічі взаємозалежних ресурсів перед їх остаточним застосуванням.

Залежно від способу застосування шаблонів для генерації веб-сторінок прийнято виділяти три основні типи WCMS-систем: *з автономною обробкою, он-лайн обробкою* і *гібридні системи.*

- 1. Автономні системи обробляють весь вміст шляхом застосуванням шаблонів перед публікацією *веб-сторінок.*
- 2. On-line системи застосовують шаблони в момент відвідування сайту користувачами (або витягають сторінки і кеша).

3. Гібридні системи комбінують перші два підходи. Деякі з них замість статичних *HTML* сторінок генерують виконувані коди *(JSP, PHP, Perl),* позбавляючи від необхідності установки WCMS-системи на кожному

Для створення динамічного сайту існують два способи. По-перше, це – написання власних програм, що відповідають за створення потрібних шаблонів і підтримують необхідні функції. При цьому система, можливо, буде повністю відповідати потребам, однак створення вимагатиме від програ містів великих зусиль і часу. Інший спосіб – це зменшення витрат часу і сил за рахунок використання існуючих систем, які називаються системами управління Web-контентом (CMC). До його недоліків можна віднести деяке зниження гнучкості, недостатній або надмірний набір можливостей.

Безперечним плюсом системи управління контентом є зниження вартості адміністрування загалом і підтримка сайту зокрема. Це відбувається за рахунок зниження втрат часу на пошуки документів, припинення дублювання і помилок, збільшення швидкості зв'язку з партнерами і клієнтами.

Основні завдання систем управління Web-контентом:

1. Розробка контенту є одним з ключових компонентів усієї системи. Саме тут починається "життєвий цикл" будь-якого матеріалу, що публікується на сайті. На цьому етапі відбувається створення, редагування і затвердження контенту, а роль системи полягає в автоматизації цих

процесів. Задача підтримки спільної роботи авторів, редакторів, програмістів і менеджерів повніс- тю перекладається на систему. Ця задача здійснюється завдяки розділенню контенту і дизайну. Всі компоненти сайту, зокрема шаблони і наповнення, зберігаються в певних місцях сховища даних. Система ж автоматично звертається до потрібних місць сховища, дозволяючи користувачам, які навіть не є технічними фахівцями, працювати над підготовкою контенту до публікації, зокрема перевіряти його достовірність

2. На рівні управління контентом розробляють сайт, попередньо переглядають і публікують підготовлений контент. Тут розробляють зовнішній вигляд, готують шаблони, розподіляють ролі користувачів і класифікують необхідну бізнес-інформацію (наприклад, товари, ціни). Важливими компонентами цього рівня є служби, які гарантують своєчасність надходження необхідного контенту

3. Доставка контенту. Коли сайт повністю підготовлений до публікації, необхідні засоби для динамічного формування Web-сторінок залежно від виду конкретних користувачів. У зв'язку з цим одним із важливих компонентів цього етапу є персоналізація або розподіл профілів, щоб кожний користувач одержував лише потрібну ту інформацію.

Необхідно зазначити, що хоч і не існує абсолютно однакових систем управління Web-контентом, експерти погоджуються в одному: із розвитком Web-технологій системи більше концентру- ватимуться на управлінні контентом, ніж на Web-паблішингу.

Системи управління Web-контентом – це програмне забезпечення, що дає змогу розробляти і підтримувати динамічні інформаційні Web-сайти. Перевага динамічних сайтів полягає у відділені дизайну від інформаційного наповнення, що дає змогу автоматизувати документообіг, бізнес - процеси,

механізми персоналізації. Системи управління Web-контентом знижують вартість створення і підтримки складних Web-серверів.

Каркасна система управління вмістом (Content Management Framework, CMF) - це інструментарій для створення систем управління вмістом, а також окремих веб-додатків. Деякі CMS, що надають API для розширення своєї функціональності, можна розглядати як CMF, наприклад WCMS *Drupal*.

На ранніх етапах розвитку Інтернету, розробка сайту зводилася до створення файлової структури з HTML-сторінок та ручного розміщення в них тексту, зображень, елементів навігації та посилань. Тоді, це не було настільки трудомістким заняттям, щоб дбати про його автоматизацію, оскільки сайти здебільшого представляли невеликі проекти, які робилися для автора чи його колег.

З часом обсяги інформації почали експоненціальне зростання, збільшилося число відвідувачів сайтів, збільшилися трудовитрати на підтримку сайту в актуальному стані. Власник сайту був змушений більшу частину часу витрачати не на безпосереднє розміщення статті або публікації, а на внесення супутньої інформації, на зразок посилань, зміни меню навігації або виправлення відомостей.

Виходом з даної ситуації стало створення спеціального класу програм – систем управління контентом, які спроможні виконували рутинні операції, не пов'язані з безпосереднім створенням статей.

В середовищі програмістів давно стало традицією те, що для кожної нової області, в якій впроваджуються програми, створюється нова мова програмування, яка буде найбільш зручною для цієї області і врахує всі її особливості. Для Інтернету на даний момент такою мовою, де-факто, є мова PHP. Її підтримують практично всі сучасні хостингові компанії, вона інтегрується з багатьма базами даних, є безкоштовною, має відкриті початкові коди і велику бібліотеку готових програм (скриптів).

В принципі і зараз існують повністю статичні сайти. Вони цілком виправдовує себе, оскільки містять максимум десяток сторінок, причому заздалегідь відомо, що сторінки не будуть змінюватися і сайт виконує лише репрезентативні функції. Переводити такий сайт на движок не доцільно.

В загальному, CMS - це програмна оболонка, яка дозволяє легко вводити і редагувати дані - текст, зображення, додавати і видаляти сторінки, тобто, керувати сайтом в режимі онлайн, без знання HTML, мов програмування та інших спеціальних навичок. CMS містить в собі комплекс найбільш поширених скриптів - наприклад, систему коментарів, голосування, фотогалерею тощо, які не доведеться шукати і додавати до сайту окремо.Оскільки створити сайт за допомогою CMS можна швидко і без спеціальних навичок, цей інструмент стає все більш популярним.

Безкоштовні системи управління контентом:

1. **WordPress**

WordPress – це найпопулярніша система управління контентом у світі, а також найпростіший спосіб створити свій власний веб-сайт чи блог. Вона має інтуїтивно-зрозумілий інтерфейс (рис. 3.3.6). та є простою у використанні. По інформації із різних джерел, WordPress використовує від 35% до 42% всіх веб-сайтів в інтернеті і цей відсоток постійно зростає. WordPress має відкритий вихідний код і поєднана з базою даних MySQL або MariaDB. WordPress дозволяє створювати сайти різного типу, інформаційні, новинні тощо, але в першу чергу, це найкращий движок для блогів.

.

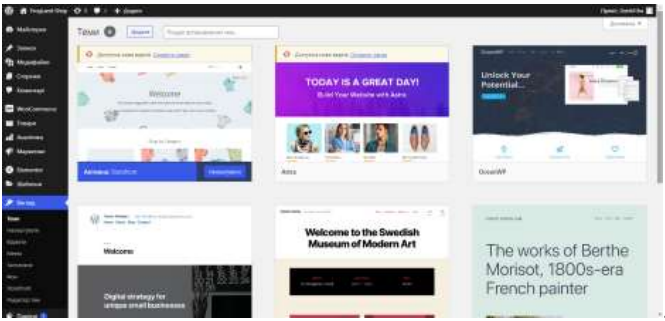


Рисунок 3.3.6 -Інтерфейс WordPress

Мінуси WordPress - не надто швидка робота сайту, можливість збоїв при високій відвідуваності та періодичне виявлення тих чи інших дірок в скриптах, потрібно налаштовувати хостинг та доменне ім'я, нести відповідальність за управління такими речами, як безпека та резервні копії. WordPress навряд чи підійде для складного сайту з великою функціональністю, порталу, Інтернет-магазину тощо - для них варто застосувати більш універсальні CMS. Вордпресс вимагає для роботи PHP та MySQL.

2. Joomla

Joomla - популярна та безкоштовна платформа CMS із відкритим кодом. В ній є безліч різних функцій, шаблонів та розширень. Для того, щоб користуватися Joomla безкоштовно, також знадобиться хостинг та ім'я домену. Інтерфейс Joomla представлено на рисунку 3.3.7.

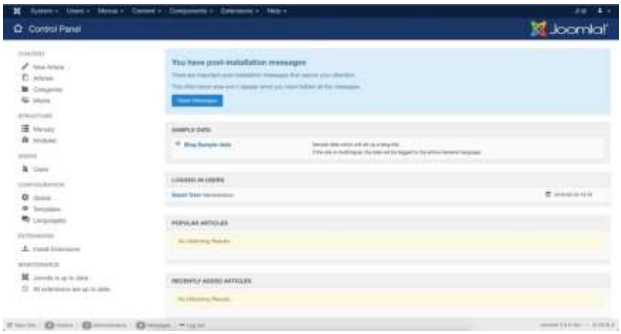


Рисунок 3.3.7 -Інтерфейс Joomla

Популярна CMS, яка є складніше в освоєнні, ніж WordPress, але має більшу сферу застосування. Для Joomla розроблено величезну кількість модулів, таких як форуми, чати, блоги, інтернет-магазини тощо, тому на ній можна сміливо робити складний багатофункціональний сайт. Для Joomla існує величезна кількість шаблонів, тому в виборі дизайну для сайту веб-розробник практично не обмежений. Joomla також має проблеми з помилками, дірками і гальмуванням сайту при великій відвідуваності.

3. Drupal

Drupal – це ще одна популярна платформа CMS із відкритим кодом.Ця CMS підійде для створення форумів, блогів (в т. ч. багатокористувацьких), онлайн-енциклопедій, сайтів спільнот. Порівнюючи Drupal з Joomla чи WordPress можна зробити висновок - ця CMS не для тих, хто хоче створити сайт легко і швидко, встановивши і відразу ж отримавши готове. Drupal зручніше для тих, хто готовий сидіти і ретельно вибудовувати структуру сайту, щоб отримати те, що йому потрібно, але не для звиклих до простих рішень новачків.

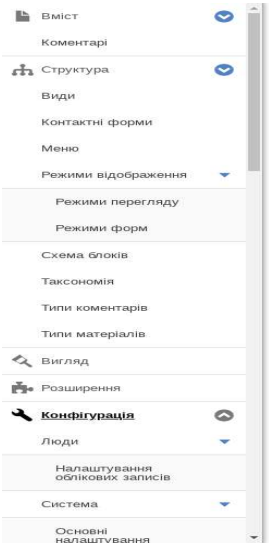


Рисунок 3.3.8 -Інтерфейс Drupal

4. Wix

Wix – також досить популярна система управління контентом, хоча вона має деякі обмеження. Wix зручний для початківців, адже дозволяє налаштовувати веб-сайт простим перетягуванням та має зручний інтерфейс (Рис. 3.3.9). Вона також постачається з ADI (Artificial Design Intelligence), що є першим в світі штучним інтелектом, який створює дизайн сайту. Wix також є безкоштовним.

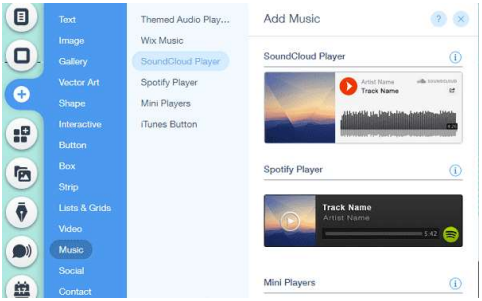


Рисунок 3.3.9 -Інтерфейс Wix

Інші безкоштовні системи:

- *PHP-nuke* - одна з найбільш старих CMS, яка раніше була досить популярною, але прославилась також і своєю слабкою захищеністю та вразливістю до зломів. Основне призначення - створення порталів, вона нескладна у встановленні і управлінні для новачків.
- *PHP-fusion* - CMS для створення порталів, має чимало модулів та шаблонів, є порівняно нескладною в установці.
- *E107* - універсальна CMS для побудови різних сайтів, має багато шаблонів і модулів. Проте деякі користувачі вказують на наявність різних недоробок і гальмування сайтів під її управлінням.
- *Mambo* - попередник Joomla, але розвивається як окремий проект.
- *Typo3* - CMS-монстр, на ній працюють багато великих сайтів, включаючи сайти відомих корпорацій і держустанов, розглядається багатьма як безкоштовний конкурент складних комерційних CMS. Велика за розміром і досить складна в освоєнні, але велика кількість можливостей може спонукати в деяких випадках зробити вибір на її користь.
- *TextPattern* - здебільшого застосовується як движок для блогів, але, подібно до WordPress, може використовуватися для створення нескладних інформаційних сайтів.

Комерційні CMS

Говорячи про відмінності комерційних CMS від безкоштовних, можна відзначити їх універсальність (переважна частина), але при цьому наявність кількох версій - від дешевих і навіть безкоштовних і простих до

більш дорогих, що містять значну функціональність і придатних для створення практично будь-якого сайту.Серед комерційних CMS можна відзначити NetCat, ABO.CMS, Amiro.CMS, UMI.CMS, Host.CMS та ін.

Окремо можна відзначити CMS Data Life Engine (DLE), яка презентована розробниками як движок для великих новинних порталів.Крім цього значна кількість веб-студій використовують при створенні сайтів універсальні CMS власної розробки.

3.3.3.Фреймворки

Фреймворки - це платформа для складних сайтів або веб-застосувань, яка оптимально підходить для створення сайтів і веб-застосувань за індивідуальним проектом, які повинні працювати швидко і витримувати серйозні навантаження. Потрібно бути готовим до того, що при виборі фреймворка в якості платформи для розробки, потрібно досить багато часу і ресурсів витратити на розробку бізнес-логіків, це плата за гнучкість.

Якщо проєкт простий , то розробники обирають CMS, а якщо проєкт складний, то простіше його робити на фреймворке .

Проєкти на базі фреймворков легко масштабовані і модернізуєми. Рішення на фреймворках працюють значно швидше і витримують більше навантаження. По рівню безпеки рішення на фреймворках значно перевершують самописні.

До проблемних питань можна віднести терміни розробки типового функціонала на фреймворках, він більший , ніж при використанні CMS. Фреймворки містять тільки базові компоненти логіки рівня додатка, тому багато функцій реалізується індивідуально. Для розробки на фреймворке потрібно чіткого розуміння процесів, які потрібно реалізувати в прєкті.

Фреймворки - Vue, Angular , React , Spring , Hibernate , Flutter – займають перші позиції в рейтингах. Кожен з них має свої відмінності, переваги та недоліки. Розглянемо їх:

3.3.3.1. Vue

Vue – фреймворк для створення користувацьких інтерфейсів, який має відкритий початковий код. Він є достатньо простим, на ньому зручно писати як односторінкові сайти так і великі проекти.

Працює як з Typescript так і з Javascript. Vue можна налаштувати під свою розробку, налаштувати структуру під власні вимоги. API цього фреймворку є досить простим в засвоєнні. Він є достатньо легким для початківців які знають тільки Javascript та HTML.

Документація до Vue написана багатьма мовами і має досить велику кількість даних які написані дуже зрозуміло для сприйняття.

Розробники Vue прив'язують представлення до відповідної моделі, фреймворк автоматично спостерігає за змінами моделі й переробляє представлення. Ця функція називається реактивність, вона робить керування станом Vue досить простим та інтуїтивно зрозумілим.

Приклад коду який написано за допомогою даного фреймворку можна побачити нижче на рис. 3.3.10.

Рисунок.3.3.10.- Приклад коду написаного за допомогою фреймворку Vue [

3.3.3.2. Angular

Angular – цей фреймворк написаний за допомогою мови Typescript. Він також має відкритий вихідний код.

Це веб-програма з відкритим кодом, очолювана Angular Team в Google та спільнотою приватних осіб та корпорацій Angular - це платформа та фреймворк для побудови односторінкових клієнтських додатків за допомогою HTML та TypeScript. Angular написаний TypeScript. Він реалізує основну та додаткову функціональність як набір бібліотек TypeScript, які ви імпортуєте у свої програми.

Архітектура програми Angular спирається на певні фундаментальні концепції. Основними будівельними елементами Angular framework є Angular компоненти, які організовані в NgModules. NgModules збирають відповідний код у функціональні набори; додаток Angular визначається набором NgModules. Додаток завжди має принаймні кореневий модуль, який дозволяє завантажувати, і, як правило, має набагато більше модулів функцій. Схема роботи фреймворку Angular представлена на рисунку 3.3.11.

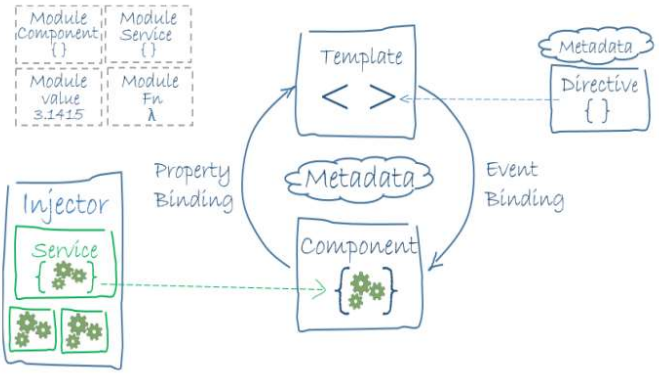


Рисунок 3.3.11. -Схема роботи фреймворку Angular

Компоненти визначають подання, які є наборами елементів екрану, які Angular може вибирати та модифікувати відповідно до логіки та даних

вашої програми. Компоненти використовують служби, які надають певні функціональні можливості, не пов'язані безпосередньо з поданнями. Постачальників послуг можна вводити в компоненти як залежності, роблячи ваш код модульним, багаторазовим та ефективним.

Модулі, компоненти та послуги - це класи, в яких використовуються декоратори. Ці декоратори позначають свій тип і надають метадані, які вказують Angular, як ними користуватися.

Метадані класу компонентів пов'язують їх із шаблоном, що визначає подання. Шаблон поєднує звичайний HTML з директивами Angular та розміткою прив'язки, які дозволяють Angular змінювати HTML перед тим, як відображати його для відображення. Метадані для класу служби надають інформацію, необхідну Angular, щоб зробити її доступною для компонентів за допомогою введення залежностей (DI).

Його основною архітектурною характеристикою є ієрархія компонентів. Він містить асинхронні компіляції шаблонів. Існує angular material — це бібліотека компонентів інтерфейсу користувача, яка реалізує Material Design в Angular. Має сервіс для заповнення автоматичного шаблону HTML, який називається Angular-language-service.

Структура та архітектура цього фреймворку створена для масштабних проектів.

Angular має велику кількість структур, що і робить його складнішим за інші фреймворки.

Також він має повільну продуктивність

3.3.3.3. React

React прекрасно підходить для створення застосунків будь-яких розмірів. Прив’язка даних є односторонньою. Що в свою чергу означає менше небажаних побічних ефектів, робить простішим створення інтерактивних інтерфейсів.

За допомогою даного фреймворку можна створити інкапсульовані

компоненти, які керують власним станом, використовуючи це відкривається можливість створення складних інтерфейсів.

Також можна просто вбудувати нові функції написані на React в старий код, що дає можливість не переписувати раніше зроблену програму. React має відкритий вихідний код.

Цей фреймворк має не тільки документацію а й посібник, за допомогою якого можна легко опанувати React.

Також можна познайомитись та спробувати експериментальні функції, яких ще не має в стабільній версії. Приклад розробки сторінки за допомогою фреймворку React наведено на рис. 3.3.12.

Рисунок 3.3.12- . Сторінка «Привіт, світе» та її код, розроблені за допомогою React.

React реалізує концепції функціонального програмування, що в свою чергу створює простий в тестуванні та багатократно використовуваний код.

При написанні коду за допомогою цього фреймворку також можна використовувати Typescript.

React відходить від компонентів на основі класів, що може бути не комфортним для тих розробників яким більше притаманне ООП.

React надзвичайно гнучкий. Саме його бібліотечний підхід зробив його таким крутим інструментом для розробки.

Розробка за допомогою цього фреймворку є дуже легкою та вільною до вибору різних компонентів.

Традиційні фреймворки, такі як Angular і Vue, покращують HTML.

Порівняння трьох фреймворків більш детально показано нижче на рис.3.3.13.

Рисунок 3.3.13 - Порівняння трьох фреймворків React, Vue та Angular

Вони використовують JavaScript всередині HTML. Вони створили атрибути HTML, які надають йому додаткові можливості.

3.3.3.4. Spring

Spring став помітним на ринку завдяки основним характеристикам Spring, якими є її модульність. Тобто його можна розділити на різні модулі, кожен з яких обслуговує свою функціональність. Spring не можна розглядати як окремий фреймворк, це скоріше сімейство фреймворків які об'єднуються у складну архітектуру:

Spring Core – основний фреймворк сімейства який відповідає за впровадження залежностей, що дозволяє розробити гнучку архітектуру

завдяки тому, що компоненти програмного додатку зв'язані між собою на рівні інтерфейсів і впроваджені в код із контейнера Spring

Spring Web MVC – надбудова над технологіями Java Servlet обраними для розробки. Фреймворк визначає основну архітектуру додатка і розділяє її на три слої (модель, вид та контролер). Це розділення важливе для відокремлення логіки додатку від генерування інтерфейсу користувача, також завдяки цьому в майбутньому буде дуже просто перейти на інший вид інтерфейсу або змінити модель без змін у інтерфейсі користувача. Також фреймворк бере на себе роботу з обробки всіх HTTP запитів клієнта.

Spring Security – у процесі роботи з клієнтами потрібно багато уваги приділяти безпеці клієнтських даних і корректній аутентифікації та авторизації користувачів, все це бере на себе Spring Security Framework. Spring Boot – фреймворк який зв'язує всі фреймворки та слугує контейнером всього додатку.

3.3.3.5. Hibernate

Hibernate – фреймворк який полегшує роботу з базою даних. Hibenate являється надбудовою над JDBC (стандарт взаємодії Java з базами даних). Він допомагає побудувати зв’язки між об’єктами Java і таблицями реляційних баз даних. Цей фреймворк не являється частиною Spring, але дуже добре вписується у сімейство і використовується разом із фреймворком Spring Data.

3.3.3.6. Flutter

Цей фрейморк дозволяє створювати нативні мобільні застосунки на основі лише однієї бази коду. Це означає, що ви можете використовувати одну мову програмування та одну кодову базу для створення двох різних додатків (для iOS та Android).

Flutter – це безкоштовний фреймворк з відкритим кодом, що був створений компанією Google та випущений у травні 2017 року як аналог

React Native від Facebook.

Flutter складається з двох важливих частин:

- SDK (Software Development Kit): набір інструментів, які допоможуть вам розробити ваші програми. Сюди входять інструменти для компіляції вашого коду в нативний машинного коду (код для iOS та Android).
- Framework (UI бібліотека на основі віджетів): набір елементів інтерфейсу користувачів (кнопки, поля введення тексту, повзунки тощо), які ви можете персоналізувати для власних потреб.

Застосунки, що створюються на основі Flutter використовуються мову програмування Dart. Мова була створена компанією Google у жовтні 2011 року. Dart – це об'єктно-орієнтована, строго типізована мова програмування. Якщо говорити про синтаксис, то це така собі суміш Java, Javascript та C# . Основна ідея побудови UI використовуючи Flutter – це побудова інтерфейсу за допомогою написання коду. Ви завжди будуєте дерево з віджетів у вашому застосунку. У вас не буде drag-and-drop інтерфейсу для додавання кнопок чи тексту на екран, який бачить юзер, натомість ви будете писати лише код.

Рис.3.3.14 - Особливості роботи Flutter

Flutter також охоплює різні платформи (Android та iOS), тобто

надає можливість створювати одночасно додаток для як для власників смартфонів на базі Android та і для власників айфонів при цьому пишучи лише один код. Працюючи над розробкою лише одного додатку ви маєте можливість створювати різні графічні інтерфейси в певних частинах застосунку, якщо в цьому є потреба. - Особливості роботи Flutter представлені на рисунку 3.3.14.

Віджетом називається абсолютно кожний елемент у Flutter. Не важливо, чи це текст, кнопка, іконка або ж навіть поле для введення тексту – усі ці елементи є віджетами.

Програмна технологія . NET Framework

.NET Framework — програмна технологія, запропонована фірмою Microsoft, як платформа для створення як звичайних програм, так і веб-застосунків. Це компонент ОС Windows, який потрібен для запуску та роботи програм, які встановлені на вашому комп'ютері.

.NET поділяється на дві основні частини— середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-коду дозволяє отримати крос-платформність на рівні скомпільованого проєкту (в термінах .NET - *збірка*), а не на рівні початкового тексту. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється у вбудованим в середовище JIT-компілятором (just in time, компіляція на льоту) в машинні коди цільового процесора.

Перехід до відкритішої моделі розробки

Революція смартфонів і планшетів внесла корективи у комп'ютерний світ. Платформа Windows перестала домінувати у світі операційних

систем, поруч з апаратною архітектурою x86 значну частку ринку зайняли інші рішення. Крос-платформовість та відкритість, які до того свідомо обмежувалася у Microsoft, вийшла на перший план і стала питанням виживання.

.NET Core

.NET Core це безкоштовний крос-платформний фреймворк з керованим кодом підтримуваний на Windows, Linux і Mac OSX. На відміну від .NET Framework вихідний код .NET Core є повністю відкритим і доступний за наступним посиланням https://github.com/dotnet/core

Він містить CoreCLR — повністю крос-платформну реалізацію CLR, віртуальну машину, яка керує виконанням програм в .NET середовищі. CoreCLR поставляється з оптимізованим «just-in-time» компілятором RyuJIT.

У той час як .NET Core розділяє підмножину API .NET Framework, він містить також власний API, який не є частиною .NET Framework.

Крім того .NET Core містить CoreRT, оптимізований під інтеграцію в AOT(компіляція перед виконанням) бінарні файли. Варіант бібліотеки .NET Core використовується для UWP (універсальна платформа Windows). UWP платформа створена Microsoft і вперше представлена в Windows 10. Метою даної платформи є допомога у створенні універсальних додатків Windows, що запускаються як на Windows 10, так і на Windows 10 Mobile без зміни в коді. Інтерфейс командного рядка .NET Core пропонує точку входу для операційних систем і надає послуги для розробників, такі як компіляція і пакети управління.

.NET Core підтримує чотири крос-платформних сценарії: ASP.NET Core веб-аплікації, консольні додатки, бібліотеки і UWP (універсальна платформа Windows) додатки.

Він не реалізує Windows Forms або WPF, які створюють стандартний графічний інтерфейс для настільних ПК на Windows.

.NET Core також модульна, а це означає, що замість збірок, розробники працюють з пакетами NuGet.

На відміну від .NET Framework, який обслуговується за допомогою служби Windows Update, .NET Core залежить від його менеджера пакетів при отриманні оновлень.

3.3.4. Технології створення мобільних додатків

Вебзастосунки або вебдодатки – це застосунки, які працюють за допомогою вебсервера, на відміну від комп'ютерних програм які запускаються локально на операційній системі приладу

Веб-додаток – це повноцінна програма яка працює в браузері, як сайт, доступ до якої здійснюється через браузер. Іншими словами, це той же сайт, тільки з інтерактивними елементами і широким функціоналом. Наприклад, Twitter, Фейсбук, YouTube та ін. За допомогою браузеру з активним мережевим підключенням користувач може відкрити вебдодаток. Електронні пошти, сайти інтернет-магазинів, онлайн-банкінг тощо – це все приклади вебзастосунків які часто використовуються користувачами

Мобільний додаток – це ті самі веб-додатки, тільки в смартфоні, що знаходяться за іконками з назвами програм у Вашому телефоні, планшеті тощо, Мобільний додаток — той, який завантажується з App Store або Google Play.

Якщо у вас на робочому столі смартфона або планшета є окрема іконка для запуску програми — це мобільна програма, яка, до речі, може частково або повністю працювати й без інтернету. Якщо для того, щоби скористатися функціями програми, потрібно зайти до Chrome, Safari або іншого вашого улюбленого браузера — це вебдодаток

Вебдодаток нічим не кращий за мобільний і навпаки. Це просто різні рішення для різних цілей і, найголовніше, дуже часто великим

проєктам потрібно й те, й інше.Щоби зрозуміти, що потрібно саме вашому проєкту, проводиться аналітика. Як і при ухваленні будь-яких інших бізнес-рішень, потрібно насамперед орієнтуватися на потреби потенційних користувачів. Цикл створення мобільних і веб-додатків дуже подібний, а результат іноді отримується ідентичний, все ж цими програмами займаються різні програмісти та використовують різні мови. Більше того, суміжні спеціалісти, що займаються проектом, також матимуть різні стратегії та підходи до роботи

Можна виокремити категорії технологій (рис.3.3.15), за допомогою яких можна створити мобільний додаток:

Рисунок 3.3.15.- Види вебзастосунків

1. Native mobile apps

1. Нативні (native mobile apps), це рідні мобільні застосунки.Цей вид додатків знаходиться безпосередньо на мобільному телефоні. Такі застосунки можна завантажити на свій пристрій за допомогою таких магазинів додатків як Google Play та App Store. Ці додатки можуть працювати від Інтернету або ж автономно.

Платформи для яких частіше за все розробляють нативні застосунки наведено на рисунку 3.3.16.

Рисунок 3.3.16.- Платформи на які частіше за все роблять native app

Native mobile apps розроблені спеціально для однієї платформи через що можуть повною мірою використовувати всі функції пристрою – камера, GPS, контакти, галерея тощо. Також вони можуть використовувати систему сповіщень та працювати в автономному режимі.

Для їх розробки використовують такі мови програмування: Java, Kotlin, Swift, Python, Swift для Android; objective-C для iOS/iPadOS. Розробка таких застосунків це дуже дорого, так як замовнику потрібно щоб додаток працював на різних платформах, що означає що потрібно розробляти декілька таких програм. Також підтримувати їх не легко, адже операційні системи постійно оновлюються, що означає що треба оновлювати і додаток теж, після виходу нової версії користувач повинен перевстановити його.

2. WEB app

Вебзастосунки – це несправжні додатки, це скоріше вебсайти, які у багатьох співвідношеннях виглядають як власні програми, але не реалізовані як такі.

Вебдодатки не треба завантажувати на пристрій, вони доступні через веббраузери. Вони виглядають як звичайні вебсайти просто можуть

адаптуватись до пристрою через який ви зайшли. Яскраво таку розробку продемонстровано на рис. 3.3.17.

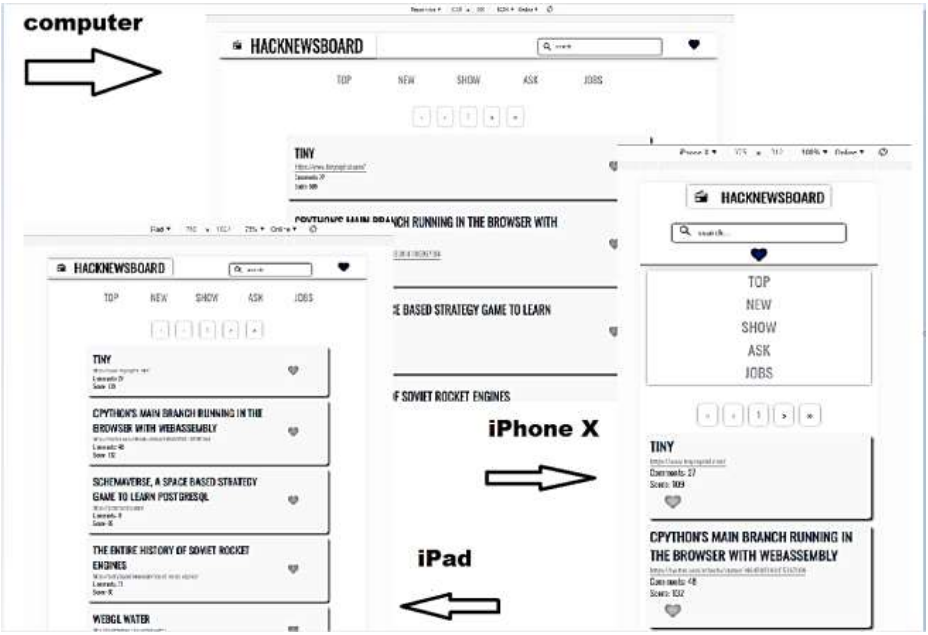


Рисунок 3.3.17 - Адаптивнсіть на прикладі вебдодатку

Для створення таких вебзастосунків зазвичай використовується HTML, Javascript, jQuery, CSS тощо. Вони можуть працювати на будь-якому пристрої, головне щоб був завантажений веббраузер.

Порівняно з вебсайтами, вебпрограми створені для взаємодії з користувачами. Вони можуть бути інформативними, але також здатні обробляти інформацію яку вносить користувач. Різницю між вебдодатком та вебсайтов можна показано на рис.3.3.18.

Рисунок 3.3.18 - Демонстрація різниці між вебдодатком та звичайним вебсайтом

Вебзастосунки надають користувачу можливість не тільки продивлятись інформацію, а й проводити маніпуляції на сторінці.

Такі взаємодії можна порівнювати з діалогом, людина взаємодіє з інтерфейсом й отримує відповідь.

Вебінтерфейс такого застосунку немає необхідності налаштовувати під різні операційні системи або платформи. Вебзастосунок можливо відкрити як на будь-якому мобільному пристрої який має браузер, так і на комп'ютері, ноутбуці чи планшеті – щоб користувач міг переглянути інформацію на всіх свої пристроях.

Також перевагою вебдодатку є те що з часом його можна прогресувати до гібридного додатку, що надасть йому ще більше переваг та функцій.

Недоліком таких застосунків є те, що вони не можуть працювати без інтернету. Залежно від браузеру можуть не відображатись ті чи інші можливості такого додатку, хоча це і може бути продумано розробником, але не завжди це можливо.

3. Hybrid app

Це гібридні додатки, їх можна завантажити на пристрій так само як і

рідні застосунки, але працюють вони як вебпрограми через браузер. Виглядають вони наче native. Гібридні застосунки можуть мати свій ярлик на домашньому екрані, високу продуктивність, чуттєвий дизайн, також можуть працювати автономно.

Це універсальні додатки, що не прив'язуються до платформи. Для їхньої розробки використовуються одночасно нативні і веб-технології. Тут можна зекономити на розробці окремо для версії iOS та Android, але такий варіант не підходить для масштабних проектів і буде менш зручним і приємним для користувачів у своїх можливостях, зручностях, функціоналі

Насправді ж це вебзастосунки які виглядають як нативні. Різницю у використанні технологій для розробки нативних та гібридних застосунків показано на рис.3.3.19.

Рисунок 3.3.19 - Демонстрація різниці технологій, які використовуються для Native та Hybrid застосунків

Гібридні додатки потребуть набагато менше коду. Швидко завантажуються, забезпечує постійний користувацький інтерфейс., інтеграція з файловими системами мобільних пристроїв.

Як і вебдодатки вони зазвичай використовують технології HTML, CSS, JavaScript тощо.

Hybrid app користуються популярність так як дозволяють

розробникам написати код один раз, але при цьому мати змогу відкривати їх на різних платформах. Механізм браузера пристрою використовується для відображення HTML та JavaScript, а також власних API-інтерфейсів для доступу до апаратного забезпечення конкретного пристрою.

Таким застосункам може невистачати потужності та шкидкості, що є відмінною характеристикою нативних застосунків.

Розробка мобільних веб-додатків

Для отримання доступу до ресурсів інтернет за допомогою тільки мобільного телефону, не вдаючись до допомоги комп'ютера або модему був розроблений спеціальний стандарт WAP.

WAP (Wire less Application Protocol) - протокол доступу до ресурсів Інтернет безпосередньо з мобільного телефону, минаючи комп'ютера та /або модем. Для розмітки документів при завантаженні їх в стільникових телефонах і інших мобільних пристроях за стандартом WAP також був розроблений і спеціальна мова - WML(Wireless Markup Language).

Спочатку WAP створювався для широкого кола технологій і стандартів бездротового мобільного зв'язку: стільникового, транкового, пейджингового та микросотовой, а також для підтримки мереж 3G. Даний стандарт інваріантний до операційного ядра з которим взаємодіє WAP-браузер і розроблявся як відкритий стандарт для бездротової передачі даних, що не залежить від постачальників пристроїв і послуг, оптимізований для мобільних телефонів, що мають дисплей з маленьким дозволом, обмеженою пам'яттю і невисокою продуктивністю.

WAP 2.0 - вдосконалена версія WAP, яка використовує урізаний варіант XHTML і CSS. Це дозволяє працювати з WAP 2.0 сайтами за допомогою звичайного браузера на комп'ютері без установки яких-небудь додаткових плагінів.

XHTML MP (XHTML Mobile Profile) - мова розмітки в WAP 2.0, розроблений для мобільних пристроїв.

Архітектура WAP аналогічна WWW. В WAP використовується той же самий спосіб адресації ресурсів і ті ж позначення типів даних. В якості клієнта виступає мобільний пристрій з вбудованим WAP-браузером, запити від якого через WAP-шлюз передаються веб-серверу, і відповідь від останнього через нього ж відправляється клієнтові.

Як сервер може виступати самий звичайний веб-сервер. У цьому випадку між WAP-шлюзом і сервером використовується протокол HTTP. З метою зменшення обсягу переданих даних, текстові ресурси, які прийшли від сервера, передаються клієнту в двійковому вигляді. Мова WML нагадує HTML, але орієнтований на пристрої з екраном низького дозволу і з невеликим розміром пам'яті. Вся інформація в WML міститься в так званих «деках».

Багато мобільні пристрої можуть показувати документи тільки в WBXML-форматі.

WBXML (WAP Binary XML) - формат компактного бінарного уявлення XML. WBXML використовується для передачі через бездротові з'єднання з низькою швидкістю. Деякі браузери зі спеціальним плагінами, дозволяють переглядати WML-сторінки на звичайному комп'ютері. В даний час спостерігається тенденція переходу від WML до XHTML.

Microsoft .NET для створення веб-додатків для мобільних пристроїв надає в розпорядження розробників інструмент .NET Mobile, що є розширення Microsoft ASP.NET і *Microsoft .NET Framework*. По суті, .NET Mobile являє собою набір серверних керуючих елементів для форм, орієнтованих на використання в бездротових мобільних пристроях. Ці елементи управління генерують різний код для різних пристроїв на мовах WML, HTML або Compact HTML(cHTML).

Загальна послідовність роботи .NET Mobile виглядає

наступним чином:

1. Мобільний пристрій. З нього виходить запит на завантаження веб-сторінки.
2. Інтернет. Запит передається через мережу відповідному веб-сервера.
3. Веб-сервер IIS п олучают запит від мобільного пристрою і передає його обробнику.
4. .NET Framework в иполняет обробку запиту.
5. ASP.NET до омпілірует запитуваний пристроєм документ.
6. .NET Mobile. Реалізує елементи веб-сторінки з урахуванням вимог конкретного типу мобільного пристрою.
7. Веб-сторінка повертається назад до клієнтського пристрою.

Сьогодні більшу частину свого життя можна помістити у компактний смартфон, адже величезний спектр потреб може задовольнити каталог додатків, що нам пропонують в Play Market або ж в Apple Store. Наступила ера мобільних застосунків

Додаток для знайомств онлайн, застосунок, що допомагає вести здоровий образ життя та підтримувати водний баланс, чи навіть банально гра-«стрілялка» - усе це може завжди бути під рукою та стати в нагоді при найменшій потребі.

Саме тому мобільні застосунки набирають такої популярності: це економить час та вирішує питання за лічені секунди роблячи наше життя простішим, цікавішим та інтерактивнішим.

Література по розділу 3

1. Інформаційні технології у суспільстві [Електронний ресурс] – Режим доступу: https://sites.google.com/site/informacijnesuspilstvo26/informacijni-tehnologiie-u-suspilstvi – Назва з екрана
2. Програмне середовище для веб-розробки [Електронний ресурс] Режим доступу: https://ospanel.io/ – Назва з екрана.
3. WordPress ВікіпедіЯ [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/WordPress – Назва з екрана.
4. phpMyAdmin ВікіпеліЯ [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/PhpMyAdmin – Назва з екрана.
5. Основні технології веб-програмування [Електронний ресурс] Режим доступу: https://www.sqroot.eu/2018/01/osnovni-tehnologiyi-veb-programuvannya/ – Назва з екрана.
6. Що таке фронтенд-розробка та як її вивчити [Електронний ресурс] Режим доступу: https://blogit.ua/shho-take-frontend-rozrobka-ta-yak-yi-vivchiti/ – Назва з екрана.
7. 10 найкращих веб-фреймворків для розробки веб-додатків [Електронний ресурс] Режим доступу: https://prostir.ua/blog/top-10-veb-frejmvorkiv-dlya-rozrobki-veb-dodatkiv/ – Назва з екрана.
8. Принципи проектування програм [Електронний ресурс] – Режим доступу до ресурсу: https://allref.com.ua/uk/skachaty/Principi_proektuvannya_program.
9. Путівник мовою програмування Python [Електронний ресурс] / Олександр Мізюк. – 2020. – Режим доступу до ресурсу: https://pythonguide.rozh2sch.org.ua
10. Соціальні мережі та месенджери в Україні [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: http://rb.com.ua/uk/blog-uk/omnibus-uk/socialni-merezhi-ta-mesendzheri-v-ukraini/.
11. Трофименко О.Г., Веб-дизайн та HTML-програмування: навч.- метод. посібник./ О. Г. Трофименко, О. Б. Козін // - Одеса: Фенікс, 2018. 194 с.
12. Астістова Т.І., Тишковець О.С., Москаленко А.М. Дослідження та розробка програмного забезпечення для мережевої системи управління комунікаціями з використанням мови програмування Java. Мехатронні системи: інновації та інжиніринг : тези доповідей IV Міжнародної науково-практичної конференції, 22 жовтня 2020 р. / Київ : КНУТД, 2020. – 228 с
13. Москаленко А.М., Аналіз та розробка програмного забезпечення для роботи з

клієнтами з використанням технології Java servlet/ А.М. Москаленко, С.М. Краснитський // Інформаційні технології в науці, виробництві та підприємництві: Збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп'ютерних наук та технологій/загал. наук. ред. В.Ю.Щербань – К.: Освіта України: 2020. –с.85 -89

14. Астістова Т.І ,Москаленко А.М. Діджиталізація гуртожитку з використанням клієнт-серверних технологій Java та фреймворка Angular // А.М. Москаленко, Б.В Науменеко, магістранти; Астістова Т.І., к.т.н., доцент /Збірник наукових праць І Всеукраїнськоїконференції «Інноватика в освт, науц та бзнес: виклики та можливост», 17 листопада 2020р., КНУТД, С . 111-119
15. Огурцов В.В. Основи веб та веб-дизайн, програмування на боці клієнта: лабораторний практикум з навчальної дисципліни "Веб-технології та веб-дизайн" для студентів напряму підготовки 6.050101 "Комп'ютерні науки"/ В.В. Огурцов , Д.В Гриньов Д.В, О.В Щербаков - Харків: ХНЕУ ім. С. Кузнеця, 2015. 208 р.
16. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools – Iuliana Cosmina,2017. – 849 с
17. 12 huge web design trends for 2018. URL: https://www.creativebloq.com/ features/web-design-trends (accessed 01.12.2018).
18. 15 Web Design Trends to Watch in 2018. URL: https://blog.hubspot.com/ marketing/web-design-trends-2017 (accessed 14.09.2018).
19. Google навчив AI розпізнавати якість фото та їхню «естетичну привабли- вість». URL: https://designtalk.club/google-navchyv-ai-rozpiznavaty-yakist-foto- i-navit-yihnyu-estetychnu-pryvablyvist/ (дата звернення 12.10.2018).
20. Marcotte E. Responsive Web design. URL: http://www.alistapart.com/ articles/responsive-web-design (accessed 26.07.2018).
21. Official Google Webmaster Central Blog: Rolling out the mobile-friendly update. URL: https://webmasters.googleblog.com/2015/04/rolling-out-mobile-friendly-update.html (accessed 26.07.2018).
22. Spring in Action. Fifth Editi on – Craig Walls October 2018 Publisher: Manning Publications, 520 pages printed in black & white,2018. – 520 с.
23. Г.Г.Швачич Сучасні інформаційно-комунікаційні технології: навчальний посібник / Г.Г.Швачич, В.В.Толстой, Л.М.Петречук, Ю.С.Іващенко, О.А.Гуляєва, О.В. Соболенко - Дніпро: НМетАУ, 2017. –230 с.

24. Пасічник О.В. Веб-дизайн. / О.В., Пасічник, В.В Пасічник – Львів: –Магнолія l, 2019 – 520с.
25. Кларенс Хо, Роб Харроп, Кріс Шефер. Spring 5 для профессионалов, 5-е издание /К.Хо, Р.Харроп, К.Шефер. // К.: Видавництво «Вильямс». – 2016. – 1120 с.
26. Діно Еспозіто, Розробка сучасних веб-додатків: аналіз предметних областей і технологій ISBN978-5-9908910-3-6, EAN-139785990891036 сторінки 464 2017 ISO/IEC/IEEE 24765:2010 Systems and software engineering — Vocabulary
27. Бейлі, Л. Вивчаємо PHP і MySQL / Л. Бейлі, М. Моррісон. - М .: Ексмо, 2014. - 800 с. - (Серія «Head First O'Reilly»).
28. Gullà F., Ceccacci S. Design Adaptable and Adaptive User Interfaces: a Method to Manage the Information // Ambient Assisted Living: Italian Forum 2014. 2014. № 3, pp.2-4.
29. Makris N. Creating Adaptable and Adaptive User Interface Implementations in Model Driven Developed Software // Radboud University Nijmegen Press. 2014. № 3,pp.8-12.
30. . Ramachandran K. Adaptive user interfaces for health care applications // IBM DeveloperWorks. 2009. № 2 (2009). С.2-5.
31. . Fowler M. Microservices - a definition of this new architectural term [Електронний ресурс] // martinfowler.com. 2014. URL: https://martinfowler.com/articles/microservices.html. Дата звернення: 28.3.2018
32. . Kim D.Reinforcement Learning-Based Dynamic Adaptation Planning Method for Architecture-based Self-Managed Software / Dongsun Kim, Sooyong Park // 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2009, pp.76-85
33. Електронний ресурс: Електронний ресурс]. Режим доступу — https://alexkosarev.name/2017/02/08/spring-framework-database-spring-data-jpa/
34. Benjamin Muschko — Gradle in Action [Електронний ресурс].— Режим доступу: https://www.manning.com/books/gradle-in-action
35. Messaging, simplified [Електронний ресурс]-Режим доступу: https://www.nexmo.com/products/sms/build?utm_source=google_search&utm_medium=paid&utm_campaign=EMEA_EURO_EN_SMS_General_B&utm_ter PHP language- Режим доступа: http://php.net/
36. Документація по Java Script. [Електронний ресурс].— Режим доступа: https://developer.mozilla.org/ua/JavaScript

37. Jquery documentation - [Електронний ресурс].— [Електронний ресурс].— Режим доступу:: http : //l query .com/
38. REST ,[Електронний ресурс] - Режим доступу:: https://uk.wikipedia.org/wiki/REST
39. Електронний ресурс Режим доступу:: https://www.historyofthings.com/history-of-the-internet
40. Rap P. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps / Rap Payne, 2019. – 309 c.
41. Gullà F., Ceccacci S. Design Adaptable and Adaptive User Interfaces: a Method to Manage the Information // Ambient Assisted Living: Italian Forum 2014. 2014. № 3, pp.2-4
42. Moore K. Flutter Apprentice // K. Moore, M. Katz, V. Ngo, 2019. – 508 c.
43. Miola A. Flutter Complete Reference: Create beautiful, fast and native apps for any device // Miola Albero, 2020. – 408 c.
44. . Makris N. Creating Adaptable and Adaptive User Interface Implementations in Model Driven Developed Software // Radboud University Nijmegen Press. 2014. № 3,pp.8-12.
45. Ramachandran K. Adaptive user interfaces for health care applications // IBM DeveloperWorks. 2009. № 2 (2009). С.2-5.
46. .Fowler M. Microservices - a definition of this new architectural term [Електронний ресурс] // martinfowler.com. 2014. URL: https://martinfowler.com/articles/microservices.html. Дата звернення: 28.3.2018
47. Kim D.Reinforcement Learning-Based Dynamic Adaptation Planning Method for Architecture-based Self-Managed Software / Dongsun Kim, Sooyong Park // 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2009, pp.76-85.
48. Windmill E. Flutter in Action // Eric Windmill, 2019. – 255 c.
49. Freitas E. Flutter Succinctly // Ed Freitas, 2019. – 129 c.
50. Mainkar P. Google Flutter Mobile Development Quick Start Guide: Get up and running with iOS and Android mobile app development // P. Mainkar, S. Giordano, 2019. – 152 c.
51. Clow M. Learn Google Flutter Fast: 65 Example Apps // Mark Clow, 2019. – 476 c.
52. Zammetti F. Practical Flutter: Improve your Mobile Developemnt with Google's Latest Open-Source SDK // Frank Zammetti, 2019. – 416 c.

53. Zaccagnino C. Programming Flutter: Native, Cross-Platform Apps the Easy Way // Carmine Zaccagnino, 2020. – 275 c.
54. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools – Iuliana Cosmina,2017. – 849 c.
55. Angular.io [Електронний ресурс]. – Режим доступу https://angular.io. 45.Spring in Action. Fifth Editi on – Craig Walls October 2018 Publisher:
56. Manning Publications, 520 pages printed in black & white,2018. – 520 c.
57. Java серверные приложения / Равиль Мухамедзянов. – К.:, 2010. –336 с.
58. Вільна інтернет-енциклопедія Wikipedia // Сі (мова програмування)[Електронний ресурс]. - НКО «Фонд Вікімедіа», 2003-2016. - Режим доступу: https://en.wikipedia.org/wiki/C_(programming_language) [дата звернення: 18.04.2018]
59. 14. Керниган, Б. Мова програмування Сі [Текст] / Б. Керниган, Д. Рітчі; пер.з анг. і ред. В. Бродові. - К .: Вільямс, 2007. - 304 с.
60. 15. Довідка по C ++ // Історія C ++ [Електронний ресурс]. – Електронна бібліотека «Cppreference.com», 2016. - Режим доступу: http://ua.cppreference.com/w/cpp/language/history [дата звернення: 18.04.2016]
61. Спільнота мови Python // Про мову Python [Електронний ресурс]. - Режим доступу: https://www.python.org/about/ [дата звернення: 18.04.2016]
62. Java і ви // Детальніше про технологію Java [Електронний ресурс]. - Режим доступу: http://www.java.com/ua /about/ [дата звернення: 19.04.2016]
63. Вязовик, Н.А. Програмування на Java [Текст] / Н.А. Вязовик ;. - 2-е вид. - К .: Інтуїт, 2016. - 600 с
64. Flutter [Електронний ресурс] // flutter.dev - 2017-2021. - Режим доступу до ресурсу: https://docs.flutter.dev/
65. What is Flutter? Benefits and limitations [Електронний ресурс] // codemagic.io. – 2018.- Режим доступу до ресурсу: https://blog.codemagic.io/what-is-flutter-benefits-and-limitations/
66. Biessek A. Flutter for Beginners / Alessandro Biessek, 2019. – 512 c.
67. Flutter [Електронний ресурс] // flutter.github.io. - 2017 - Режим доступу до ресурсу: https://flutter.github.io/samples/#
68. TypeScript https://en.wikipedia.org/wiki/TypeScript
69. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools –Iuliana Cosmina,2017. – 849 c.

70. Spring in Action. Fifth Editi on – Craig Walls October 2018 Publisher:
71. Manning Publications, 520 pages printed in black & white,2018. – 520 с.
72. Nadeem and M. Y. Javed, “A performance comparison of data encryption algorithms,” International Conference on Information and Communication Technologies, pp. 84–89, 2005
73. Астістова Т.І Тишковець О.С., , Дослідження та розробка програмного забезпечення для мережевої системи управління комунікаціями з використанням мови програмування Java / О.С Тишковець , Т.І. Астістова // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектування. – К. : Освіта України, 2020. – С156-159
74. Астістова Т. І.,Потапенко М.О, Аналіз та розробка програмного забезпечення e-commerce системи з розподіленим навантаженням / Т.І. Астістова , М.О. Потапенко // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектування. – К. : Освіта України, 2020. – С159-162
75. Астістова Т.І., Підгайний М.О., Дослідження та розробка математичного забезпечення для управління контентом на основі технологій SPRING та HIBERNATE/ Т. І Астістова, М.О., Підгайний // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектування. – К. : Освіта України, 2020. – С 232 – 236
76. 16. Астістова Т.І. Аналіз програм та сервісів для поширення сайтів в інтернет-просторі / Т. І Астістова // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектування. – К. : Освіта України, 2020. – С170-175
77. Астістова Т. І., Глембоцький В.С., Програмне забезпечення для системи ідентифікації студентів// Т.І. Астістова, В.С.Глембоцький // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедрикомп’ютерних наук та технологій. – К. : Освіта України, 2021 р. – С 208 – 211
78. Астістова Т. І., Кольва М. А., Розробка інтерфейсу системи управління розумним будинком / Т.І. Астістова , М.А.Кольва // Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених,

аспірантів, магістрів кафедри комп'ютерних наук та технологій. – К. : Освіта України, 2021 р. – С. 212 – 214

79. Астістова Т. І., Егоров Д.С., Розробка QR-коду для веб додатку «Система ідентифікації студентів»//Т.І. Астістова, Д.С.Егоров// Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри комп'ютерних наук та технологій. – К. : Освіта України, 2021 р. . – С. 214 – 217
80. Грицюк Ю. І., Аналіз вимог до програмного забезпечення. Навчальний посібник/ Ю. І. Грицюк .// Київ,2018, С.425
81. Майкл Віттіг. «Amazon Web Services in Action ». США, Manning, 1237 – 1469 с
82. .AstistovaTetyana, Development of software for accounting for the presence of student attendance// T.I. Astistova, D. D Liakhovska D. D., M. V.Nykytyuk, N.V.Smorzhevsky // Збірник наукових праць VI Mтждународная научно-практическая конференція «Science, innovations and education: problems and prospects» , 13-15 января 2022 года.Токіо, Японія, С.111-116, сертифікат
83. AstistovaTetyana, Development of a web-system for monitoring climate control indicators// T.I. Astistova, D.M. Kochuk, Y.O. Karas, Y.V. Sezko// Збірник наукових праць *VII* Международная научно-практическая конференція «International scientific innovations in human life»*19-21* января 2022 года, Манчестер, Великобритания, С. 134-136,
84. Керниган, Б. Мова програмування Сі [Текст] / Б. Керниган, Д. Рітчі; пер.з анг. і ред. В. Бродові. - М .: Вільямс, 2007. - 304 с.
85. Спільнота мови Python // Про мову Python [Електронний ресурс]. – Python Software Foundation,. - Режим доступу: https://www.python.org/about/ [дата звернення: 18.04.2016]
86. Ріхтер, Дж. CLR via C #. Програмування на платформі Microsoft .NET Framework 4.0 мовою C # [Текст] / Дж. Ріхтер; пер. з анг. І. Радченко та І. Рузмайкіна, зав. ред. А. Кривцов, рук. пр. А. Юрченко, вед. ред. Ю. Сергієнко, літ. ред. А. Жданов, кор. В. Лістова,. - 3-edbl/. – Київ.: Пітер, 2012. – 928 с.
87. Гослінг, Д. Мова програмування Java SE 8. Детальний опис [Текст] / Д. Гослінг, Б. Джой, Г.Л. Стіл, Г. Брача, А. Баклі; пер. з анг. І. Карася, зав. ред. С.Н. Тригуб, літ. ред. Л.Н. Красножон, худ. ред. В.Г. Павлютін, кор. Л.А. Гордієнко, вер. М.А. Удалов. - 5-е изд. - К.: Вільямс, 2015. - 672 с.
88. Крістіансен, Т. Програмування на Perl [Текст] / Т. Крістіансен, б. д фої, Л. Уолл,

Дж. Орвант; пер. з анг. А. Кисельова, , 2014. - 1048 с. - (Серія «Head First O'Reilly»).

89. Штайн, Л. Розробка мережевих програм на Perl [Текст] / Л. Штайн; пер. з анг. К. Птіцина. – Київ Вільямс .:, 2001. - 752 с.
90. Бейлі, Л. Вивчаємо PHP і MySQL [Текст] / Л. Бейлі, М. Моррісон. -К.:Новий світ , 2010. - 800 с. - (Серія «Head First O'Reilly»).
91. Provos, Niels; Mazières, David; Talan Jason Sutton 2012 (1999). "A Future-Adaptable Password Scheme". Proceedings of 1999 USENIX Annual Technical Conference: 81–92.
92. Nadeem and M. Y. Javed, “A performance comparison of data encryption algorithms,” International Conference on Information and Communication Technologies, pp. 84–89, 2005
93. Schneier, “Description of a new variable-length key, 64-bit block cipher (Blowfish),” Proc. Fast Softw. Encryption Cambridge Secur. Work. Cambridge, U. K., pp. 191–204, 1994.

4. ЕЛЕМЕНТИ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

4.1. Дослідження операцій – кількісна перспектива прийняття рішень

Знання, інновації та технології змінюються, і тому прийняття рішень у сучасному соціальному та бізнес-середовищі стало складним завданням через невелику кількість або відсутність прецедентів. Висока вартість технологій, матеріалів, робочої сили, тиск конкуренції та безліч економічних, соціальних, політичних факторів і точок зору значно ускладнили процес прийняття управлінських рішень. Для ефективного вирішення ширших тактичних і стратегічних питань, а також для забезпечення лідерства в глобальному бізнес-середовищі особи, які приймають рішення, не можуть дозволити собі приймати рішення на основі свого особистого досвіду, здогадок або інтуїції, оскільки наслідки неправильних рішень можуть виявитися серйозними та дорогий. Наприклад, вихід на неправильні ринки, виробництво неправильних продуктів, надання неналежних послуг тощо може спричинити серйозні фінансові проблеми для організацій. Отже, особам, які приймають рішення, бажано розуміти використання кількісних методів для прийняття рішень. Небагато осіб, які приймають рішення, стверджують, що підхід АБО не відповідає належним чином потребам бізнесу та промисловості. Невиконання висновків є однією з основних причин. Серед причин невдачі впровадження – нездатність особи, яка приймає рішення, творчо розв’язувати проблеми. Процес реалізації передбачає, що етапи визначення, аналізу, моделювання та вирішення проекту виконано відповідно до встановлених інструкцій. Підхід до дослідження операцій допомагає порівнювати всі можливі альтернативи (шляхи дій або дії) щодо їхніх потенційних результатів, а потім аналізувати чутливість рішення до змін або помилок у числових значеннях. Однак цей підхід (або техніка) є допоміжним для суджень осіб, які приймають рішення, а не заміною його. Намагаючись розв’язати проблему реального життя, особа, яка приймає

рішення, повинна розглянути цю проблему як з кількісної, так і з якісної точки зору. Для прикладу розглянемо проблему інвестицій у три альтернативи: фондовий ринок, нерухомість і банківський депозит. Щоб прийняти будь-яке рішення, інвестор повинен вивчити певні кількісні фактори, такі як фінансові показники з балансів компаній, акції яких розглядаються; грошові потоки компаній з нерухомості та норми прибутку від інвестицій у нерухомість; і скільки коштуватимуть інвестиції в майбутньому, якщо їх покласти в банк за певною відсотковою ставкою на певну кількість років? Крім того, певні якісні фактори, такі як погодні умови, державна та центральна політика, нові технології, політична ситуація тощо? Оцінка кожної альтернативи може бути надзвичайно складною або займати багато часу з двох причин: по-перше, кількість і складність інформації, яку потрібно обробити, і, по-друге, наявність великої кількості альтернативних рішень. З цих причин особи, які приймають рішення, все частіше звертаються до кількісних факторів і використовують комп'ютери для досягнення оптимального рішення проблем. Існує потреба в структурному аналізі з використанням операційних досліджень/кількісних методів, щоб прийти до цілісного вирішення будь-якої управлінської проблеми. Це можна зробити шляхом критичного вивчення рівнів взаємодії між прикладним процесом дослідження операцій та різними системами організації. Це створює концептуальну основу організаційних/управлінських структур і процесу застосування дослідження операцій.

Чисельні методи оптимізації є основою підходів до дослідження операцій. Діяльність менеджера або інженера при дослідженні задач дослідження операцій (створенні нових зразків техніки) складається з трьох основних етапів: синтез, аналіз, оптимізація, які поєднані у послідовності. На першому етапі синтезу визначаються базові принципи побудови нового пристрою (створюваної технічної системи) або умови

прийняття керуючого рішення та його структура. На другому етапі аналізу, як правило, за допомогою побудованої моделі визначаються залежності властивостей пристрою або керуючого рішення від параметрів, які можна змінювати. Останній етап оптимізації має метою визначення таких значень параметрів, при яких керуюче рішення має найкращі у певному розумінні властивості. У роботі розглядаються графічні методи для ілюстрації методів, що застосовуються для безумовних задач нелінійного математичного програмування на етапі оптимізації. Прийняття рішень у сучасному соціальному та бізнес-середовищі є складним завданням через велику кількість або відсутність прецедентів. Галузь науки дослідження операцій розглядає вплив навколишнього середовища разом із організаційними структурами та управлінською поведінкою для прийняття рішень. Загальновідомо, що дослідження операцій виникло як дисципліна під час другої світової війни, коли виникла гостра потреба в управлінні обмеженими ресурсами. Однак конкретну модель і техніку ОР можна простежити ще під час Першої світової війни, коли Томас Едісон (1914–1915) спробував використати тактичну ігрову дошку для пошуку рішення мінімізації втрат від підводних човнів противника, замість того, щоб ризикувати кораблями в реальних умовах війни. Приблизно в цей же час датський інженер А. К. Ерланг проводив експерименти з вивчення коливань попиту на телефонне обладнання з використанням автоматичного набору номера. Пізніше такі експерименти були використані як основа для розвитку теорії черги очікування. Оскільки Друга світова війна пов'язана зі стратегічними і тактичними проблемами, які були дуже складними, очікувати адекватних рішень від окремих осіб чи спеціалістів однієї дисципліни було нереально. Особи, які в сукупності вважалися фахівцями з математики, економіки, статистики та теорії ймовірностей, інженерії та фізичних наук, були сформовані як спеціальні підрозділи в збройних силах для вирішення стратегічних і тактичних

завдань різних військових операцій. Такі групи спочатку були сформовані ВПС Великобританії, а пізніше американські збройні сили сформували подібні групи. Одна з груп у Британії стала відома як «Цирк Блекетта». Ця група під керівництвом професора П. М. С. Блекетта була приєднана до підрозділу радіолокаційних оперативних досліджень і їй було доручено аналізувати координацію радіолокаційного обладнання на стрілецьких пунктах. Після успіху цієї групи подібний підхід до змішаних команд був також прийнятий в інших союзних країнах. Після Другої світової війни вчені, які брали активну участь у військових групах АБО, доклали зусиль, щоб застосувати підхід дослідження операцій до цивільних проблем, пов’язаних з бізнесом, промисловістю, дослідженнями тощо. Наступні три чинники лежать в основі оцінки використання дослідження операцій підхід: (і) Економічний і промисловий бум призвів до механізації, автоматизації та децентралізації операцій і розподілу функцій управління. Ця індустріалізація призвела до складних управлінських проблем, і тому застосування дослідження операцій до прийняття управлінських рішень стало популярним. Продовження досліджень після війни призвело до прогресу в різних методах дослідження операцій. У 1947 році Дж. Б. Данціг розробив концепцію лінійного програмування, розв’язок якої знаходить за допомогою методу, відомого як симплекс-метод. Окрім лінійного програмування, до 1950-х років було добре розроблено багато інших методів АБО, таких як статистичний контроль якості, динамічне програмування, теорія масового обслуговування та теорія запасів. Використання високошвидкісних комп'ютерів дозволило застосувати методи АБО для розв'язання реальних проблем прийняття рішень.

Наступні три чинники лежать в основі оцінки використання дослідження операцій підхід:

Економічний і промисловий бум призвів до механізації, автоматизації та децентралізації операцій і розподілу функцій управління.

Ця індустріалізація призвела до складних управлінських проблем, і тому застосування дослідження операцій до прийняття управлінських рішень стало популярним.

Продовження досліджень після війни призвело до прогресу в різних методах дослідження операцій.

Широкий спектр застосувань дослідження операцій спонукав різні організації та окремих осіб визначати його таким чином: z Дослідження операцій — це застосування наукових методів до складних проблем у керуванні й управлінні великими системами людей, машин, матеріалів і грошей у промисловості, бізнесу, уряду та оборони. Особливий підхід полягає в розробці наукової моделі системи, що включає вимірювання таких факторів, як випадковість і ризик, за допомогою яких можна передбачити та порівняти результати альтернативних рішень, стратегій або засобів контролю. Мета полягає в тому, щоб допомогти керівництву науково визначити свою політику та дії. – (Operational Research Society, Великобританія) Застосування наукового методу до вивчення операцій великих складних організацій або діяльності. Це забезпечує адміністраторів найвищого рівня кількісною основою для рішень, які підвищать ефективність таких організацій у виконанні їхніх основних цілей. – Національна дослідницька рада комітету з ОР, США. Визначення, надане Товариством операційних досліджень Великобританії, було розкритиковане через наголос, який він приділяє складним проблемам і великим системам, залишаючи у читача враження, що це високотехнічний підхід, придатний лише для великі організації. Дослідження операцій – це систематичне застосування кількісних методів, технік та інструментів для аналізу проблем, пов’язаних з роботою систем. – Daellenbach and George, 1978 Дослідження операцій – це, по суті, набір математичних методів та інструментів, які в поєднанні з системним підходом застосовуються для вирішення практичних проблем економічного чи інженерного характеру. –

Daellenbach and George, 1978. Ці два визначення передбачають інше бачення АБО – сукупності моделей і методів, які були розроблені в основному незалежно один від одного. Дослідження операцій використовує плановий підхід (оновлений науковий метод) і міждисциплінарну команду для представлення складних функціональних зв'язків у вигляді математичних моделей з метою забезпечення кількісної основи для прийняття рішень і виявлення нових проблем для кількісного аналізу.

Ця сфера прийняття рішень (Дослідження операцій) характеризується використанням наукових знань через міждисциплінарну роботу команди з метою визначення найкращого використання обмежених ресурсів. – Н А Taha, 1976 Ці два визначення відносяться до міждисциплінарної природи ОР. Однак ніщо не може завадити одній людині розглянути кілька аспектів проблеми, що розглядається. z Дослідження операцій, у найзагальнішому розумінні, можна охарактеризувати як застосування наукових методів, технік та інструментів до проблем, пов'язаних з функціонуванням системи, щоб забезпечити тим, хто контролює операції, оптимальні рішення проблем. – Churchman, Ackoff and Arnoff, 1957 Це визначення відноситься до дослідження операцій як до техніки вибору найкращого курсу дій з кількох доступних варіантів дій, щоб досягти бажаного вирішення проблеми. Дослідження операцій описано як метод, підхід, набір технік, командна діяльність, поєднання багатьох дисциплін, розширення окремих дисциплін (математики, інженерії, економіки), нова дисципліна, покликання, навіть релігія. Можливо, це деякі з усіх цих речей. – С. Л. Кук, 1977 р. Дослідження операцій можна описати як науковий підхід до прийняття рішень, який передбачає роботу організаційної системи. – Ф. С. Гіллер і Г. Дж. Ліберман, 1980 р. Дослідження операцій – це науковий метод забезпечення виконавчих відділів кількісною основою для прийняття

рішень щодо операцій, які вони контролюють. – П.М. Морзе та Г.Е. Кімбол, 1951 z Дослідження операцій – це прикладна теорія прийняття рішень. Він використовує будь-які наукові, математичні чи логічні засоби, щоб спробувати впоратися з проблемами, з якими стикається виконавча влада, коли вона намагається досягти повної раціональності у вирішенні своїх проблем прийняття рішень. – Д. В. Міллер і М. К. Стар, 1969 р. z Дослідження операцій – це науковий підхід до вирішення проблем виконавчого керівництва. – Г. М. Вагнер У міру розвитку дисципліни дослідження операцій їй було надано численні назви, такі як аналіз операцій, системний аналіз, аналіз рішень, наука управління, кількісний аналіз, наука прийняття рішень. Це пояснюється тим фактом, що типи проблем, які виникають, завжди пов'язані з «ефективним рішенням». Дослідження операцій — це застосування наукових методів, прийомів та інструментів до проблем, пов'язаних із функціонуванням систем, щоб забезпечити тим, хто контролює операції, оптимальні рішення проблем (Черчмен та ін.). Науковий метод полягає у спостереженні та визначенні проблеми; формулювання та перевірка гіпотези; та аналіз результатів тесту. Отримані таким чином дані потім використовуються для прийняття рішення про те, чи слід прийняти гіпотезу чи ні. Якщо гіпотеза прийнята, результати повинні бути реалізовані, інакше ні. В більшості випадків задача дослідження операцій формулюється у вигляді задачі нелінійного програмування.

Найбільш поширеними моделями дослідження операцій є моделі лінійного програмування. Побудова моделей лінійного програмування ґрунтується на припущенні про лінійну залежність між характеристиками об'єкта та елементами рішення. Таке припущення у технологічних дослідженнях далеко не завжди виявляється виправданим. Можна вказати багато об'єктів оптимізації, у тому числі й у деревообробці, для яких воно є неправомірним. Нелінійними, як правило, є залежності техніко-

економічних, силових та якісних характеристик процесів механічної обробки деревини від режимних факторів. Модель завдання про оптимальне розміщення виробництва також виявляється при уважному розгляді грубої, оскільки передбачає лінійну залежність між витратами виробництво та його обсягом Насправді ця залежність має складніший характер (рис. 4.1.1). На початковій ділянці I вона випукла нагору. Це тим, що з зростанням випуску витрати виробництва одиниці виробленої продукції знижуються. Потім, досягши деякого обсягу випуску, подальше його збільшення досягається ціною навантаження виробництва, що призводить до збільшення собівартості продукції (ділянка II).

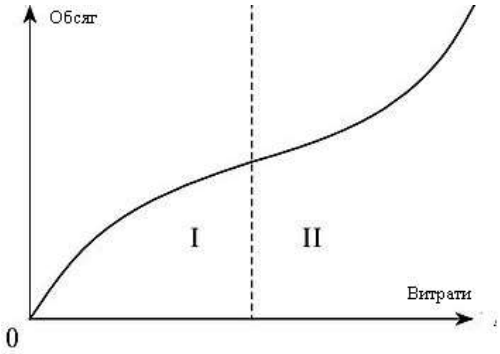


Рис 4.1.1 Залежність між витратами на виробництво та його обсягом

Широкий спектр застосувань дослідження операцій спонукав різні організації та окремих осіб визначати його таким чином: Дослідження операцій — це застосування наукових методів до складних проблем у керуванні й управлінні великими системами людей, машин, матеріалів і грошей у промисловості, бізнесу, уряду та оборони. Особливий підхід полягає в розробці наукової моделі системи, що включає вимірювання таких факторів, як випадковість і ризик, за допомогою яких можна передбачити та порівняти результати альтернативних рішень, стратегій або

засобів контролю. Мета полягає в тому, щоб допомогти керівництву науково визначити свою політику та дії. – Operational Research Society, Великобританія Застосування наукового методу до вивчення операцій великих складних організацій або діяльності. Це забезпечує адміністраторів найвищого рівня кількісною основою для рішень, які підвищать ефективність таких організацій у виконанні їхніх основних цілей. – Національна дослідницька рада комітету з ОР, США. Визначення, надане Товариством операційних досліджень Великобританії, було розкритиковане через наголос, який він приділяє складним проблемам і великим системам, залишаючи у читача враження, що це високотехнічний підхід, придатний лише для великі організації. Дослідження операцій – це систематичне застосування кількісних методів, технік та інструментів для аналізу проблем, пов'язаних з роботою систем. – Daellenbach and George, 1978 Дослідження операцій – це, по суті, набір математичних методів та інструментів, які в поєднанні з системним підходом застосовуються для вирішення практичних проблем економічного чи інженерного характеру. – Daellenbach and George, 1978. Ці два визначення передбачають інше бачення АБО – сукупності моделей і методів, які були розроблені в основному незалежно один від одного. Дослідження операцій використовує плановий підхід (оновлений науковий метод) і міждисциплінарну команду для представлення складних функціональних зв'язків у вигляді математичних моделей з метою забезпечення кількісної основи для прийняття рішень і виявлення нових проблем для кількісного аналізу. – Thierauf and Klekamp, 1975

Ключовою особою у післявоєнному розвитку ОР був Джордж Б. Данціг

В Індії дослідження операцій виникло в 1949 році, коли в Регіональній дослідницькій лабораторії в Гайдарабаді було створено підрозділ операційної операції для планування та організації досліджень.

Дослідження операцій є галуззю управлінської науки, яка займається вивченням та вдосконаленням процесів прийняття рішень в організаціях. В Україні дослідження операцій розвивається і має свою історію.

Початки дослідження операцій в Україні можна відслідкувати ще з радянських часів. У 1960-1970-х роках були створені перші наукові колективи та лабораторії, що займалися дослідженням операцій. Ці дослідження використовувалися для покращення ефективності виробництва в різних промислових галузях, в тому числі в енергетиці, транспорті та інших секторах економіки.

Після отримання незалежності Україною в 1991 році дослідження операцій продовжили свій розвиток. Були створені спеціалізовані науково-дослідні інститути, вузи розпочали викладати дисципліни з дослідження операцій, а також створювалися консалтингові компанії, які надавали послуги з оптимізації бізнес-процесів та управління ризиками.

Сьогодні в Україні дослідження операцій застосовуються в різних сферах, таких як логістика, фінанси, маркетинг, громадський транспорт, енергетика та багато інших. Завдяки методам дослідження операцій можна оптимізувати робочі процеси, підвищити ефективність виробництва та прийняття рішень, знизити витрати та ризики.

Дослідження операцій пов’язане з науковим вирішенням того, як найкраще проектувати та експлуатувати людино-машинні системи, які зазвичай вимагають розподілу обмежених ресурсів. Ця сфера прийняття рішень характеризується використанням наукових знань через міждисциплінарну роботу команди з метою визначення найкращого використання обмежених ресурсів. – Н А Taha, 1976 Ці два визначення відносяться до міждисциплінарної природи ОР. Однак ніщо не може завадити одній людині розглянути кілька аспектів проблеми, що розглядається. Дослідження операцій, у найзагальнішому розумінні, можна охарактеризувати як застосування наукових методів, технік та

інструментів до проблем, пов'язаних з функціонуванням системи, щоб забезпечити тим, хто контролює операції, оптимальні рішення проблем. – Churchman, Ackoff and Arnoff, 1957 Це визначення відноситься до дослідження операцій як до техніки вибору найкращого курсу дій з кількох доступних варіантів дій, щоб досягти бажаного вирішення проблеми. Дослідження операцій описано як метод, підхід, набір технік, командна діяльність, поєднання багатьох дисциплін, розширення окремих дисциплін (математики, інженерії, економіки), нова дисципліна, покликання, навіть релігія. Можливо, це деякі з усіх цих речей. – С. Л. Кук, 1977 р. Дослідження операцій можна описати як науковий підхід до прийняття рішень, який передбачає роботу організаційної системи. – Ф. С. Гіллер і Г. Дж. Ліберман, 1980 р. Дослідження операцій – це науковий метод забезпечення виконавчих відділів кількісною основою для прийняття рішень щодо операцій, які вони контролюють. – П.М. Морзе та Г.Е. Кімбол, 1951 Дослідження операцій – це прикладна теорія прийняття рішень. Він використовує будь-які наукові, математичні чи логічні засоби, щоб спробувати впоратися з проблемами, з якими стикається виконавча влада, коли вона намагається досягти повної раціональності у вирішенні своїх проблем прийняття рішень. – Д. В. Міллер і М. К. Стар, 1969 р. Дослідження операцій – це науковий підхід до вирішення проблем виконавчого керівництва. – Г. М. Вагнер У міру розвитку дисципліни дослідження операцій їй було надано численні назви, такі як аналіз операцій, системний аналіз, аналіз рішень, наука управління, кількісний аналіз, наука прийняття рішень. Це пов'язано з тим, що типи проблем є різними.

4.2. Підхід дослідження операцій до рішення проблем

Найважливішою особливістю дослідження операцій є використання наукового методу та побудова моделей рішень. Підхід дослідження

операцій до вирішення проблем базується на трьох фазах, а саме (i) Фаза судження; (ii) Фаза дослідження та (iii) Фаза дій.

Фаза судження Ця фаза включає: (i) ідентифікацію реальної проблеми, (ii) вибір відповідної цілі та значень різних змінних, пов'язаних із цією метою, (iii) застосування відповідної шкали вимірювання, тобто прийняття рішення щодо міри ефективності (бажаності) і (iv) формулювання відповідної моделі проблеми та абстрагування істотної інформації, щоб можна було отримати рішення для досягнення цілей особи, яка приймає рішення.

Фаза дослідження Ця фаза є найбільшою та найдовшою серед усіх фаз. Однак, незважаючи на те, що решта два не такі довгі, вони також однаково важливі, оскільки забезпечують основу для наукового методу. На цьому етапі використовуються: (i) спостереження та збір даних для кращого розуміння проблеми, (ii) формулювання гіпотези та моделі, (iii) спостереження та експерименти для перевірки гіпотези на основі додаткових даних, (iv) аналіз доступної інформації та перевірка гіпотези з використанням попередньо встановлених показників бажаності, (v) передбачення різних результатів гіпотези та (iv) узагальнення результату та розгляд альтернативних методів.

Фаза дій Ця фаза складається з надання рекомендацій щодо виконання рішення. Це рішення реалізується особою, яка в змозі реалізувати результати. Ця особа повинна знати про середовище, в якому виникла проблема, усвідомлювати мету, припущення, що стоять за проблемою, і необхідні пропуски (узагальнення) моделі.

Моделі не відображають і не можуть представляти кожен аспект реальної проблеми/системи через її великі та мінливі характеристики. Проте модель можна використовувати для аналізу, розуміння та опису певних аспектів (ключових особливостей) системи з метою покращення її продуктивності, а також для вивчення змін (якщо такі є), не порушуючи

поточні операції. Наприклад, щоб вивчити потік матеріалів у виробничій фірмі, можна побудувати масштабовану діаграму, що показує фабрику, розташування обладнання, інструментів і працівників. Немає необхідності вказувати деталі що не мають відношення до виробництва.

Ключ до побудови моделі полягає в абстрагуванні лише відповідних змінних, які впливають на критерії показників ефективності даної системи, і у вираженні зв'язку у відповідній формі. Однак модель повинна бути максимально простою, щоб дати бажаний результат. З іншого боку, надмірне спрощення проблеми також може призвести до неправильного рішення. Збагачення моделі відбувається шляхом зміни значення змінних і послаблення припущень. Основними трьома якостями будь-якої моделі є: • Валідність моделі – модель повинна відображати критичні аспекти досліджуваної системи/проблеми, • Зручність використання моделі – модель може бути використана для конкретних цілей, і • Цінність модель для користувача. Окрім цих трьох якостей, іншими факторами, що представляють інтерес, є (і) вартість моделі та її складність, (іі) час, витрачений на формулювання моделі тощо. Неформальне визначення моделі, яке стосується всіх, є інструментом для мислення та розуміння особливостей будь-якої проблеми/системи перед тим, як діяти. Наприклад, модель, як правило, формулюється, коли (а) ми думаємо про те, що хтось скаже у відповідь на наші дії, (б) ми намагаємося вирішити, як витратити наші гроші, або (в) ми намагаємося передбачити наслідки якоїсь діяльності (нашої, чиєїсь чи навіть природної події). Іншими словами, ми не зможемо вивести або здійснити будь-яку цілеспрямовану дію, якщо спочатку не сформуємо модель діяльності. Підхід АБО використовує цю природну тенденцію до створення моделей. Ця тенденція змушує більш ретельно і ретельно думати про моделі, які ми збираємося використовувати. Загалом моделі ж різні класифікатори, але базовими і корисними для методів дослідження є наступні.

Символічні моделі Ці моделі використовують алгебраїчні символи (літери, цифри) і функції для представлення змінних і їхніх зв'язків для опису властивостей системи. Такі відносини також можна представити у фізичній формі. Символічні моделі є точними й абстрактними, їх можна аналізувати за допомогою законів математики.

Символічні моделі поділяються на дві категорії.

Вербальні моделі Ці моделі описують властивості системи письмовою чи усною мовою. Письмові речення, книги тощо є прикладами словесної моделі.

Математичні моделі Ці моделі використовують математичні символи, літери, цифри та математичні оператори (+, –, ÷, ×) для представлення зв'язків між змінними системи для опису її властивостей або поведінки. Рішення таких моделей отримують шляхом застосування відповідної математичної техніки. Методи дослідження найважливіших математичних моделей визначають математичні методи дослідження операцій.

Математичні методи дослідження операцій є одним з важливих підрозділів дослідження операцій. У дослідженні операцій теж розглядається задача пошуку умовного екстремуму функції, але, на відміну від математичного програмування, вигляд обмежень задачі не обмежується рівнянням або нерівністю типу « ≤ » чи « ≥ ».

Розділ математики, що називається математичним програмуванням, безпосередньо пов'язаний з питаннями оптимізації. Зміст математичного програмування складають теорія і методи розв'язання задач про пошук мінімального або максимального значення функції на множені значень аргументів, що задовольняють сукупності рівнянь і нерівностей.

Задачу математичного програмування формулюють таким чином. Подані функція $F(x_1, x_2, \ldots, x_n)$ n дійсних змінних і m співвідношень вигляду $G_1(x_1, x_2, \ldots, x_n) \{\leq, \geq, =\} 0$, $G_2(x_1, x_2, \ldots, x_n) \{\leq, \geq, =\} 0$, … , $G_m(x_1, x_2,$

$. . . , x_n) \{\leq, \geq, =\}\ 0$. Треба визначити максимальне або мінімальне значення функції F при умові, що змінні $x_1, x_2, . . . , x_n$ задовольняють усім m співвідношенням. (У фігурних дужках наведені можливі знаки співвідношення).

Задача математичного програмування найчастіше записується у такій формі

$$F(x_1, x_2, . . . , x_n) \to max\ (min),$$

$$f_i(x_1, x_2, . . . , x_n) \{\leq, \geq, =\}\ 0,\ i = 1, \ldots , m.$$

Функція F, $max\ (min)$ якої треба знайти, називається цільовою функцією (функцією цілі), а сукупність співвідношень

$f_i(x_1, x_2, . . . , x_n) \{\leq, \geq, =\}\ 0,\ i = 1, m$ - обмеженнями задачі. Максимум або мінімум надалі будемо називати екстремумом або екстремальним значенням.

Зручніше формулювати так:

знайти мінімум (або максимум) функції

$$f(\boldsymbol{x}), \boldsymbol{x} \in E^n, \qquad (4.2.1)$$

за виконання умов

$$fi(\boldsymbol{x})\ ОП_i\ 0,\ i=1,...,m, \qquad (4.2.2)$$

де $ОП_i$ одна з операцій порівняння $ОП_i \in \{\leq, =, \geq\}$.

Кожний упорядкований набір значень змінних $\boldsymbol{x}$, що задовольняє обмеженням, називається допустимим розв'язком задачі або планом задачі. Задача може мати багато допустимих розв'язків. Множина усіх допустимих розв'язків називається допустимою множиною задачі або областю допустимих розв'язків (допустимою областю).

Допустимий розв'язок, на якому цільова функція приймає шукане екстремальне значення, називається оптимальним розв'язком (планом) задачі або просто розв'язком. Значення цільової функції на будь-якому оптимальному плані називається оптимумом цієї задачі.

Не кожна задача математичного програмування має розв'язок. По-перше, це має місце, коли не існує жодного набору значень змінних, що задовольняють обмеженням. У цьому разі кажуть, що допустима множина пуста. По-друге, задача може не мати розв'язку навіть, коли допустима множина не пуста, але цільова функція на цій множині не обмежена. Наприклад, задача

$$tg\ x \rightarrow max,$$
$$x \le \pi/2,$$
$$x \ge 0;$$

розв'язку не має, оскільки не існує такого значення x, при якому $tg\ x$ приймає максимальне значення на відрізку $[0, \pi/2]$. Дійсно, для кожного x_1 такого, що $0 < x_1 < \pi/2$, можна вказати x_2 ($x_1 < x_2 < \pi/2$), для якого $tg\ x_2 > tg\ x_1$, а у точці $\pi/2$ $tg\ x$ не існує. Графічна ілюстрація наведена на рис. 4.2.2.

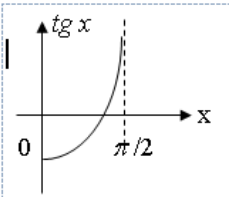


Рис. 4.2.2. Поведінка цільової функції *tg*.

Отже задача математичного програмування може не мати розв'язку у двох випадках: 1) допустима область пуста; 2) цільова функція не обмежена.

4.3. Приклади побудови математичної моделі

4.3.1. Задача про розподіл ресурсів

Є підприємство, що випускає n видів продукції і при цьому застосовує m видів сировини. Кожний вид продукції має певну ціну: c_j – ціна j-го виду продукції, $j = 1, \ldots, n$; є певний запас сировини кожного виду: b_i

- запас i-го виду сировини, $i = 1, \dots, m$. При виготовленні одиниці продукції j-го виду витрати сировини i-го виду дорівнюють a_{ij}; витрати усіх видів сировини при виготовленні одиниці кожного виду продукції визначаються матрицею $A = ||a_{ij}||_{i,j=1}^{m,n}$. Вважається, що технологія лінійна, тобто витрати сировини пропорційні об'єму випуску продукції. Треба визначити об'єми випуску кожного виду продукції, при яких сумарна вартість випущеної продукції максимальна.

Побудова моделі, починається з визначення змінних. Найчастіше змінні позначають шукані величини. Нехай x_j - об'єм випуску j-го виду продукції, $j = 1, \dots, n$. Тоді цільова функція визначається, як

$$F = \sum_{j=1}^{n} c_j x_j = (c, x) \to max,$$

де $c = (c_1, c_2, \dots, c_n)$, $x = (x_1, x_2, \dots, x_n)$.

Врахуємо запаси і витрати сировини. Витрати i-го виду сировини для випуску продукції j-го виду визначаються як $a_{ij}x_j$. Тому витрати сировини i-го виду на випуск продукції усіх видів дорівнює $a_{ij}x_j$, а врахування запасів усіх видів сировини висловлюється системою нерівностей

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1, \dots, m,$$

або у матричній формі

$$Ax \le b,$$

де $b = (b_1, b_2, \dots, b_m)$.

Враховуючи, що об'єми випуску продукції невід'ємні, остаточно отримуємо модель у вигляді

$$F = (c, x) \to max,$$

$$Ax \le b,$$

$$x \ge 0.$$

Остання нерівність являє скорочений запис системи

$$\begin{cases} x_1 \geq 0; \\ x_2 \geq 0; \\ \quad \ldots \\ x_n \geq 0. \end{cases}$$

Розглянута вище постановка задачі відповідає плануванню роботи підприємства за валом.

Часто буває необхідно враховувати і асортимент продукції, що випускається.

4.3.2. Задача про оптимальне завантаження устаткування (нелінійна модель)

Задачі нелінійного програмування займають особливе місце серед усіх задач оптимізації. Перш за все, в цьому класі існують задачі визначення мінімального та максимального значення функції аргумент яких може приймати значення в усьому просторі *En*, без обмежень. Це неможливо для задач лінійного програмування та лінійного дискретного програмування. Ще деякі з ряду причин, що це обумовлюють, наведені нижче.

1. На відміну від задач лінійного програмування, вирішення задачі нелінійного програмування, навіть користуючись необхідними стандартними програмами, може одержати тільки спеціаліст, що розуміє властивості задач, і особливості реалізації алгоритмів різноманітних методів, що можуть бути використані. Результат, звичайно, одержують комбінуючи декілька алгоритмів. Правильна послідовність застосування цих алгоритмів може виявитися вирішальним чинником одержання розв'язання.

У задачі лінійного програмування чинником, що звичайно, лімітує, є тільки розмірність. Завершення алгоритму призводить до ясної, конкретної ситуації - отримано розв'язання або розв'язання не існує. Причин для зупинки програми, що реалізує алгоритм НЛП істотно більше.

2.Моделі нелінійного програмування часто найбільше вдала поступка між адекватним описом реальної задачі і практичної можливості одержати вирішення. Наприклад, одна з найпоширеніших задач ДО - задача про оптимальне завантаження устаткування. Звичайно цю задачу формулюють у вигляді задачі лінійного програмування. Історія лінійних моделей для цієї задачі починається з робіт Канторовича і Леонтьева. Проте, для багатьох практично важливих випадків, використання лінійної моделі призводить до абсурдних результатів. Покажемо це на прикладі.

Задача про оптимальне завантаження устаткування формулюється в такий спосіб.

Є n засобів виробництва (станків, технологій або технологічних ліній). Засоби пронумеровані $j=1,\ldots ,n$. Вважається, що відома об'ємна виробнича програма, якщо задана інтенсивність x_j (час роботи устаткування, довжина тканини, що необхідно перетворити в сорочки і т.д. - прийнятий термін) застосування кожного засобу виробництва. Надалі, для скорочення викладень ми будемо вважати, що інтенсивність - це час роботи устаткування. У лінійній моделі передбачається, що кількість виробленої продукції i - того вигляду є лінійною функцією інтенсивності застосування кожного засобу виробництва і дорівнює $a_{ij}x_j$. Символом a_{ij} позначимо кількість продукції i- того виду, що буде зроблено j- тим засобом виробництва при використанні його з одиничною інтенсивністю (за одиницю часу). Сумарне виробництво всіма способами продукції i-того виду дорівнює

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, \ldots, m,$$

або у матричній формі

$Ax = b,$

а витрати на виробництво, пов'язані із застосуванням відповідного способу також рівні $c_i x_i$. Сумарні витрати $\sum_{j=1}^{n} c_j x_j$.

У цьому випадку модель задачі формулюється в такий спосіб - необхідно визначити інтенсивність застосування кожного способу виробництва (вектор *X*), при якому витрати на виробництво (c,x) - (скалярний добуток двох векторів) будуть мінімальними, а кількість зробленої продукції буде не менше необхідної

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m,$$

або у матричній формі

$Ax \geq b, x_j \geq 0.$

Якщо одним з обмежень є витрати часу на виробництво (плановий інтервал часу не більше *T*), то відповідне обмеження буде мати вигляд

$$\sum_{j=1}^{n} x_j \leq T$$

Ця модель підкуповує простотою, але не враховує час переналагодження устаткування.

У більшості реальних ситуацій для того, щоб реалізувати необхідний спосіб виробництва, устаткування потрібно зупинити й переналагодити. Якщо $t(x_j)$ - час, необхідний для організації j - того способу виробництва, то реальний час, протягом якого устаткування буде виробляти продукцію T - $\sum_{j=1}^{n} t(x_j)$. Підсумовування здійснюється тільки по тим компонентам вектора *X*, що більше 0. Функція $t(x_j) = 0$ якщо $x_j = 0$ і $t(x_j) = t_j$ якщо $x_j > 0$. З огляду на властивості задачі лінійного програмування можна припустити, що відмінних від нуля компонент у векторі вирішення задачі *X* буде r= min (n,m) (m-рівно кількості обмежень). Якщо m і n достатньо великі числа, то час, протягом якого устаткування буде виробляти продукцію відповідно до нашої програми, може бути менше 0.

Цю перешкоду легко перебороти за допомогою нелінійної або дискретної моделі, проте відомі дискретні моделі цієї задачі будуть мати

таку розмірність, що їхній реальний аналіз неможливий, а вирішення нелінійної задачі є цілком реальним.

Витрати на переналагодження обладнання є одноразовими і займають фіксований час. Реальна функція витрат на реалізацію j - того способу виробництва виглядає в такий спосіб $c_j(x_j)= 0$, якщо $x_j = 0$, $c_j(x_j)= c_{pj}$, якщо $0 < x_j < t_j$, $c_j(x_j)= c_{pj}+c_j*(x_j - t_{pi})$, якщо $x_j > t_{pi}$

Ця функція має розрив в точці $x_j = 0$. Тут c_j можна інтерпретувати як приведені витрати на реалізацію і- того способу виробництва в одиницю часу, а c_{pi} як витрати на переналагодження устаткування з метою реалізації j того способу виробництва.

Виробництво продукції j тим способом, з урахуванням переналагоджень можна описати виразом $a_{ij}(x_i) = 0$, якщо $0 < x_i < t_i$, $a_{ij}(x_i) = ap_{ij}(x_j - t_j)$ якщо

$x_j \geq t_j$. Ця функція не має похідних в точці $x_j = a$ але є безперервною.

Тепер адекватна модель задачі планування завантаження устаткування має вид

$$\sum_{j=1}^{n} c_j(x_j) \text{ -> min}$$

$$\sum_{j=1}^{n} a_{ij}(x_i) \geq b_i , \ i = 1, \ldots , \ m,$$

$$\sum_{j=1}^{n} x_j \leq T$$

$x_j \geq 0$..

T - тривалість інтервалу планування

Ця задача є типовою задачею дискретного програмування, тобто належить колу задач, для яких не відомі ефективні алгоритми вирішення (для відомих алгоритмів кількість обчислень для вирішення задачі є експоненціальною функцією від кількості змінних). Задача є задачею дискретного програмування тому, що функція $c_j(x_j)$ є розривною.

Нижче приведена задача нелінійного програмування яка є еквівалентною задаче 1, але може бути досліджувана більш ефективними методами, що гарантують успіх. Ця задача відрізняється лише виглядом цільової функції. Модифікована функція витрат на реалізацію j - того способу виробництва виглядає в такий спосіб $c_j(x_j) = x_j * (c_{pj} / t_j,)$ якщо $0 < x_j < t_j,$, $c_j(x_j) = c_{pj} + c_j * (x_j - t_{pi})$, якщо $x_j > t_{pi}$

Задача з модифікованою функцією витрат є еквівалентною задачі дискретного програмування (розв'язок нелінійної задачі є розв'язком дискретної), є задачею з безперервною цільовою функцією і для неї можуть бути застосовані алгоритми нелінійного програмування.

4.3.3. Задача планування роботи підприємства з урахуванням асортименту продукції, що випускається (цілочисельна модель)

Подана задач являє собою попередню, доповнену додатковими обмеженнями, що висловлюють той факт, що продукція повинна випускатися комплектами, які містять певні кількості одиниць продукції кожного виду: d_j - кількість одиниць j-го виду продукції у комплекті, $j = 1, \dots, n$ $(d = (d_1, d_2, \dots, d_n))$. Треба визначити об'єми випуску кожного виду продукції, при яких кількість випущених комплектів максимальна.

Для побудови математичної моделі, крім змінних $x_j, j = 1, \dots, n$, уведемо змінну y, що дорівнює шуканій кількості випущених комплектів. Тоді цільова функція приймає вигляд

$F = y \to max.$

Додаткові обмеження, що відображають вимогу комплектності продукції, можна записати у вигляді

y - ціле, $x_j \geq y d_j, j = 1, \dots, n.$

Остаточно модель має вигляд

$F = y \to max,$

$Ax \leq b,$

$x \geq yd$

$x \geq 0,$

y - ціле.

4.4. Основні принципи дослідження лінійних моделей

4.4.1 Евклідовий простір

Сукупність усіх упорядкованих послідовностей n дійсних чисел, для якої визначені операція додавання будь-яких двох послідовностей та множення кожної послідовності на дійсне число, називається n-вимірним лінійним векторним простором.

Кожна послідовність з n-вимірного лінійного векторного простору позначається як $A = (a_1, a_2, \dots, a_n)$ і називається вектором або точкою простору.

Числа $a_j, j = \overline{1, n}$ називають компонентами (координатами) вектору A або координатами точки A.

Сумою векторів $A = (a_1, a_2, \dots, a_n)$ і $B = (b_1, b_2, \dots, b_n)$ називають вектор $C = (c_1, c_2, \dots, c_n)$, координати якого визначаються таким чином $c_j = \overline{a_j + b_j}, j = 1, n$.

Вектор $A = (a_1, a_2, \dots, a_n)$ дорівнює вектору $B = (b_1, b_2, \dots, b_n)$, якщо $a_j = b_j, \overline{j = 1, n}$.

Вектор $\overline{0} = (0, 0, \dots, 0)$ називають нульовим вектором. Надалі для вилучення непорозумінь, де можливе двозначне тлумачення, над позначенням вектору будемо застосовувати риску.

Добутком вектора $A = (a_1, a_2, \dots, a_n)$ на дійсне число k називають вектор $B = (b_1, b_2, \dots, b_n)$, координати якого визначаються так $b_j = \overline{ka_j}, j = 1, n$.

Вектор $(-1)A = -A = (-a_1, -a_2, \dots, -a_n)$ називають протилежним вектору A.

Різницею двох векторів A та B називають вектор, що визначається так $A - B = A + (-B)$.

Скалярним добутком двох векторів $A = (a_1, a_2, \dots, a_n)$ та $B = (b_1, b_2, \dots, b_n)$ називають дійсне число, що позначається як (A, B) і визначається за формулою $(A, B) = \sum_{j=1}^{n} a_j b_j$.

Довжиною вектора A або його модулем називають дійсне число, що позначається як $|A|$ і обчислюється таким чином $|A| = \sqrt{(A, A)} = \sqrt{\sum_{j=1}^{n} a_j^2}$.

Кут φ між векторами A та B визначається згідно з формулою

$$\varphi = \arccos \frac{(A, B)}{|A|\,|B|}.$$

Лінійна залежність векторів

Вираз вигляду $\sum_{j=1}^{r} k_j A_j$ називається лінійною комбінацією векторів $A_1, A_2, \dots, A_r$.

Система (сукупність) векторів $A_1, A_2, \dots, A_r$ $(r \geq 2)$ називається лінійно залежною, якщо існують такі числа $k_1, k_2, \dots, k_r$, серед яких не усі дорівнюють нулю, що має місце рівняння

$$\sum_{j=1}^{r} k_j A_j = \bar{0}. \qquad (3.1)$$

Якщо вказані числа не існують, система векторів називається лінійно незалежною. Для лінійно незалежної системи векторів $A_1, A_2, \dots, A_r$ з рівняння (3.1) прямує $k_j = 0$ для усіх j.

Максимальна кількість лінійно незалежних векторів, що належать поданій системі, називається рангом системи векторів.

Ранг матриці – порядок максимальної квадратної її підматриці, визначник якої не дорівнює нулю.

Теорема 3.1. (Про ранг системи векторів). Нехай подана система векторів $V = \{ a_1, a_2, \dots, a_m \}$:

$$a_1 = (a_{11}, a_{12}, \dots, a_{1n}),$$
$$a_2 = (a_{21}, a_{22}, \dots, a_{2n}),$$
$$\cdot \quad \cdot \quad \cdot$$
$$a_m = (a_{m1}, a_{m2}, \dots, a_{mn}).$$

Побудуємо матрицю A на координатих цих векторів як на елементах:

$$A = \begin{pmatrix} a_{11} & a_{12} \dots & a_{1n} \\ a_{21} & a_{22} \dots & a_{2n} \\ . & . & . \\ a_{m1} & a_{m2} \dots & a_{mn} \end{pmatrix}$$

Тоді ранг матриці A дорівнює рангу системи векторів V: rank(V) = rank(A).

Доведення. Спочатку доведемо, що rank(V) rank(A). Нехай rank(A) = r. Це означає, що матриця A містить квадратну підматрицю B порядку r, визначник якої не дорівнює нулю. Без втрати загальності можна припустити, що ця підматриця розташована у верхньому лівому куту, оскільки цієї умови завжди можна досягти, змінивши порядок векторів у системі та порядок координат векторівОскільки визначник матриці B не дорівнює нулю (|B| 0), перші r рядків матриці A лінійно незалежні. Дійсно, якби ці рядки були лінійно залежні, то і рядки матриці B були лінійно залежні, що суперечить умові |B| 0.

$$A = \begin{pmatrix} \overset{B}{\downarrow} & & & & & \\ a_{11} & a_{12} \dots & a_{1r} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} \dots & a_{2r} & \dots & a_{2j} & \dots & a_{2n} \\ . & . & . & & . & & \\ a_{r1} & a_{r2} \dots & a_{rr} & \dots & a_{rj} & \dots & a_{rn} \\ . & . & . & & . & & \\ a_{i1} & a_{i2} \dots & a_{ir} & \dots & a_{ij} & \dots & a_{in} \\ . & . & . & & . & & \\ a_{m1} & a_{m2} \dots & a_{mr} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix}$$

Тепер доведемо, що rank(V) rank(A). Для цього покажемо, що коли додати до перших r рядків матриці A довільний рядок цієї матриці, то отримаємо лінійно залежну систему рядків.

Розглянемо квадратну підматрицю B' матриці A порядку r+1, що утворена матрицею B, доповненою елементами довільного j-го стовпця та довільного i-го рядка.

$$B' = \begin{pmatrix} a_{11} & a_{12} \dots & a_{1r} & a_{1j} \\ a_{21} & a_{22} \dots & a_{2r} & a_{2j} \\ . & . & . & \\ & & & \end{pmatrix}$$

$$a_{r1}\ \ a_{r2}\ \ \dots\ a_{rr}\ a_{rj}$$
$$a_{i1}\ \ a_{i2}\ \ \dots\ a_{ir}\ a_{ij}$$

Визначник матриці B' дорівнює нулю, оскільки $\text{rank}(A) = r$. Розкладемо його за елементами j-го стовпця:

$$a_{1j}A_{1j} + a_{2j}A_{2j} + \dots + a_{rj}A_{rj} + a_{ij}A_{ij} = 0,$$

де A_{sj} $(s = \overline{1,\ r},\ s = i)$ – відповідні алгебраїчні доповнення. Оскільки $A_{ij} \neq 0$, маємо

$$a_{ij} = -a_{1j}A_{1j}/A_{ij} - a_{2j}A_{2j}/A_{ij} - \dots - a_{rj}A_{rj}/A_{ij} =$$
$$= k_1a_{1j} + k_2a_{2j} + \dots + k_r a_{rj},$$

де $k_s = -A_{sj}/A_{ij}$, $s = \overline{1,\ r}$.

Підставляючи у матрицю B' послідовно елементи усіх стовпців матриці A $(j = \overline{1,\ n})$, отримуємо

$$a_{i1} = k_1a_{11} + k_2a_{21} + \dots + k_r a_{r1},$$
$$a_{i2} = k_1a_{12} + k_2a_{22} + \dots + k_r a_{r2},$$
$$\cdot \quad \cdot \quad \cdot$$
$$a_{in} = k_1a_{1n} + k_2a_{2n} + \dots + k_r a_{rn}.$$

Останню систему рівнянь можна записати у вигляді

$$a_i = k_1a_1 + k_2a_2 + \dots + k_r a_r,$$

тобто i-й рядок є лінійною комбінацією перших r рядків, і цим завершується доведення.

Оскільки при транспонуванні матриці ранг її не змінюється, то максимальна кількість лінійно незалежних стовпчиків у матриці A також дорівнює r.

Наслідок. Будь-яка система, що містить n+1 n-вимірних векторів, лінійно залежна.

Дійсно, побудувавши на компонентах цих векторів як на елементах матрицю A, отримуємо матрицю розміру n+1 x n, ранг якої, очевидно, менше n+1.

Базис n-вимірного лінійного векторного простору

Базисом n-вимірного лінійного векторного простору називається сукупність n лінійно незалежних векторів цього простору.

Теорема 3.2. (Про розкладання вектору за базисом.) Кожний вектор n-вимірного лінійного простору можна записати у вигляді лінійної комбінації базисних векторів і тільки одним чином.

Доведення. Нехай $E_1, E_2, \dots, E_n$ - базис. Візьмемо довільний вектор A. Система векторів $E_1, E_2, \dots, E_n, A$ лінійно залежна. Тому $k_0A + k_1E_1 + k_2E_2 + \dots + k_nE_n = 0$, де не усі k_j ($j = 0, n$) дорівнюють нулю. Число k_0 не дорівнює нулю, оскільки інакше вектори $E_1, E_2, \dots, E_n$ були б лінійно залежні. Тому

$$A = (-k_1/k_0)E_1 + (-k_2/k_0)E_2 + \dots + (-k_n/k_0)E_n, \quad (3.2)$$

тобто A – лінійна комбінація векторів базису.

Тепер доведемо, що розкладання за поданим базисом тільки одне. Припустимо, що крім розкладання (3.2), існує інше

$$A = s_1E_1 + s_2E_2 + \dots + s_nE_n. \quad (3.3)$$

Після віднімання від (3.3) рівняння (3.2) маємо

$$0 = (s_1+k_1/k_0)E_1 + (s_2+k_2/k_0)E_2 + \dots + (s_n+k_n/k_0)E_n.$$

Оскільки лінійні комбінації (3.2) і (3.3) відрізняються принаймні одним коефіцієнтом, останнє рівняння суперечить лінійній незалежності базисних векторів, і цим завершується доведення..

Лінійний векторний простір може мати багато базисів. Серед них виділяють один - ортонормований базис:

$$e_1 = (1, 0, \dots, 0),$$
$$e_2 = (0, 1, \dots, 0),$$
$$\cdot \quad \cdot \quad \cdot$$
$$e_n = (0, 0, \dots, 1).$$

Ці вектори, очевидно, лінійно незалежні. Довжина кожного дорівнює одиниці, а скалярний добуток довільних двох різних векторів дорівнює

нулю, і це означає, що кут між ними становить /2. Коефіцієнти розкладання будь-якого вектору за ортонормованим базисом є компонентами цього вектору

$$A = (a_1, a_2, \ldots a_n) = a_1e_1 + a_2e_2 + \ldots + a_ne_n.$$

Тепер повернемося до аналізу методу Жордана-Гауса. У результаті застосування методу вихідна матриця A коефіцієнтів системи рівнянь перетворюється. Позначимо сукупність стовпчиків, що відповідають вилученим змінним, як матрицю B. В результаті застосування методу Жордана-Гауса до сумісної системи стовпчики матриці B перетворюються у одиничні стовпчики. Сукупність цих одиничних стовпчиків утворює одиничну матрицю з, можливо, переставленими стовпцями. Отримати таку матрицю можна шляхом множення матриці B на обернену B^{-1} і потім на перестановочну матрицю P. Перестановочна матриця містить елементи, що дорівнюють нулю або одиниці, і у кожному рядку та у кожному стовпчику міститься тільки одна одиниця. Оскільки множення матриць асоціативне, можна вважати, що у результаті виконання методу Жордана-Гауса матриця B помножається на матрицю $B^{-1\sim} = B^{-1} P$. А це, у свою чергу, означає, що і уся матриця A помножається на $B^{-1\sim}$. Умовно систему Ax = b можна записати у вигляді [B R]x = b. Умовність полягає у тому, що у матриці A стовпці матриці B можуть довільним чином чергуватися із стовпцями матриці R. Результат перетворення системи можна записати у вигляді $[B^{-1\sim} B\ B^{-1\sim}R]\ x = B^{-1\sim}b$.

Існування для матриці B оберненої матриці B^{-1} означає, що її стовпці лінійно незалежні і тому утворюють базис у просторі стовпців (m-вимірних векторів, якщо матриця B має порядок m). Саме це виправдовує раніше застосований термін “базисні змінні”, оскільки ці змінні відповідають базисним стовпчикам.

Вихідну систему рівнянь можна уявити у вигляді $\sum_{j=1}^{n} a^j \sum_{j=1}^{n} x_j = b$, де a^j - j-й стовпець матриці A. Таким чином шукані змінні – це коефіцієнти

лінійної комбінації стовпців матриці A, що дорівнює вектору правих частин рівнянь. Базисний розв'язок являє коефіцієнти розкладання вектору b за базисними стовпцями. Якщо усі базисні змінні не дорівнюють нулю, то базисний розв'язок називають невиродженим. У іншому разі базисний розв'язок вироджений. Геометричний смисл виродження полягає у тому, що вектор b є лінійною комбінацією деякої власної підмножини множини базисних стовпців.

Форма системи рівнянь, отримана після застосування метода Жордана-Гауса, називається приведеною. Кожному базису у просторі стовпців відповідає своя приведена форма. Всього таких форм існує стільки, скільки різних базисів можна утворити зі стовпців матриці A. Якщо ранг матриці A дорівнює m, кількість стовпців – n, то максимальна можлива кількість різних базисів дорівнює $C^m_n = n!/(m!(n-m)!)$. У конкретному випадку кількість базисів може виявитися меншою, оскільки не усі сукупності з m стовпців можуть бути лінійно незалежними.

4.4.2 ФОРМУЛЮВАННЯ ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Задача лінійного програмування може бути сформульована у декількох формах.

Загальна форма має вигляд

$$F = \sum_{j=1}^{n} c_j x_j \to \max (\min),$$

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i, \quad \overline{i = 1, m_1},$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad \overline{i = m_1+1, m_2},$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, \quad \overline{i = m_2+1, m},$$

$$x_j \geq 0 \quad j = 1, n.$$

Останні $n_2 - n_1+1$ обмеження мають тільки по одній змінній і називаються прямими обмеженнями. Попередні m обмежень, що можуть містити довільну кількість змінних називають непрямими. Особливість

загальної форми полягає у тому, що непрямі обмеження можуть бути як рівняннями, так і нерівності обох типів (“ ” чи “ “), а також у тому, що на довільну кількість змінних може накладатися умова невід’ємності.

Канонічна форма записується у вигляді

$$F = \sum_{j=1}^{n} c_j x_j \rightarrow \max (\min),$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = \overline{1, m},$$

$$x_j \geq 0 \quad j = \overline{1, n}.$$

У канонічній формі усі непрямі обмеження мають вигляд рівнянь, і на усі змінні накладається умова невід’ємності.

4.4.3 Еквівалентні перетворення задачі лінійного програмування

Дві задачі будемо називати еквівалентними, якщо розв’язок будь-якої з них дає можливість очевидним чином отримати розв’язок іншої.

В загальному випадку задача максимізації $f(\boldsymbol{x})$ еквівалентна задачі подібній задачі мінімізації $-f(\boldsymbol{x})$, а нерівність $fi(\boldsymbol{x}) \geq 0$ можна замінити $-fi(\boldsymbol{x}) \leq 0$, рівність $fi(\boldsymbol{x})=0$ можна замінити

$$f_i(\boldsymbol{x}) \leq 0 \quad \text{і} \quad -f_i(\boldsymbol{x}) \leq 0.$$

Від будь-якої форми задачі лінійного програмування завжди можна перейти до еквівалентної задачі у будь-якій іншій формі. Наведемо операції перетворення, які при цьому застосовуються.

1. Від пошуку максимуму цільової функції завжди можна перейти до пошуку мінімуму. Дійсно, якщо у деякій точці x^0 функція F досягає максимуму, то функція -F у точці x^0 досягає мінімуму. Це можна умовно висловити у такій формі

$$F(x) \rightarrow \max \Leftrightarrow -F(x) \rightarrow \min.$$

Таким чином, помноживши вираз для цільової функції на -1, можна перейти від пошуку максимуму до пошуку мінімуму і навпаки.

2. Зміст обмеження-нерівності завжди можна змінити на протилежний, помноживши нерівність на –1.

$$\sum_{i=1}^{n} a_{ij}x_j \geq b_i \Leftrightarrow \sum_{j=1}^{n} -a_{ij}x_j \leq -b_i.$$

3. Від обмеження-нерівності типу « » завжди можна перейти до еквівалентної система, що складається з обмеження-рівняння і додаткового прямого обмеження Уведена змінна x_{n+i} називається додатковою на відміну від основних змінних $x_1, x_2, \dots, x_n$.

4. Аналогічно здійснюється перетворення обмеження-нерівності типу ≤

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \Leftrightarrow \begin{cases} \sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i; \\ x_{n+i} \geq 0. \end{cases}$$

5. Від обмеження-рівняння завжди можна перейти до системи двох нерівностей.

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \Leftrightarrow \begin{cases} \sum_{j=1}^{n} a_{ij}x_j \geq b_i; \\ \sum_{j=1}^{n} a_{ij}x_j \leq b_i. \end{cases}$$

Помноживши одну з нерівностей на -1, можна отримати систему нерівностей одного типу.

6. Від задачі із змінними довільного знаку завжди можна перейти до еквівалентної задачі, усі змінні якої невід'ємні. Припустимо, змінна x_k у вихідній задачі може мати довільний знак. Тоді, виконавши підстановку $x_k = x'_k - x''_k$, де x'_k, x''_k 0, у вираз для цільової функції та у обмеження, отримаємо еквівалентну задачу, кількість змінних з довільним знаком якої зменшено на одиницю. Виконавши аналогічні підстановки для усіх

змінних з довільним знаком, отримаємо еквівалентну задачу, усі змінні якої невід'ємні. Допустимість вказаної підстановки прямує з того, що різниця двох невід'ємних величин може мати довільний знак.

4.4.4. Системи лінійних рівнянь

Оскільки у більшості методів розв'язування задачі лінійного програмування використовується канонічна форма, що являє собою систему рівнянь для невід'ємних змінних, розглянемо властивості систем лінійних рівнянь. Систему лінійних рівнянь можна уявити у вигляді

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad \overline{i = 1, m,}$$

або у матричній формі $Ax = b$, де матриця коефіцієнтів $A = \|a_{ij}\|_{i,j=1}^{m,n}$.

Розв'язками системи називається сукупність значень змінних $x_1 = d_1$, $x_2 = d_2$, … , $x_n = d_n$, яка задовольняє усім рівнянням системи. Система, що має хоча б один розв'язок називається сумісною, а система, що не має жодного розв'язку називається несумісною. Сумісна система, що має тільки один розв'язок називається визначеною. Система що має більше одного розв'язку називається невизначеною. Дві системи називаються рівносильними, якщо множини їх розв'язків співпадають. Під розв'язуванням системи лінійних рівнянь розуміють визначення множини усіх розв'язків системи.

Метод Жордана-Гауса

У основі чисельного методу розв'язування системи лінійних рівнянь лежить теорема про перетворення системи лінійних рівнянь.

Теорема 2.1. Якщо і-те рівняння системи замінити сумою цього рівняння з j-м рівнянням, помноженим на дійсне число t, то отримана система рівносильна вихідній.

Доведення. Нехай $d = (d_1, d_2, \dots, d_n)$ - розв'язок вихідної системи. Це означає, що

$$\sum_{k=1}^{n} a_{ik}d_k = b_i,$$

$$\sum_{k=1}^{n} a_{jk}d_k = b_j.$$

Помноживши друге числове рівняння на число t і додавши добуток до першого, отримуємо

$$\sum_{k=1}^{n} a_{ik}d_k + \sum_{k=1}^{n} ta_{jk}d_k = b_i + tb_j = \sum_{k=1}^{n}(a_{ik} + ta_{jk})d_k.$$

Звідси маємо, що d задовольняє рівнянню

$$\sum_{k=1}^{n}(a_{ik} + ta_{jk})x_k = b_i + tb_j,$$

і тому є розв'язком нової системи.

Тепер припустимо, що $d = (d_1, d_2, \dots, d_n)$ - розв'язок нової системи. Це означає, що мають місце такі числові рівняння

$$\sum_{k=1}^{n}(a_{ik} + ta_{jk})d_k = b_i + tb_j,$$

$$\sum_{k=1}^{n} a_{jk}d_k = b_j.$$

Помноживши друге рівняння на $-t$ і додавши добуток до першого, переконуємося, що d задовольняє i-му рівнянню вихідної системи і є її розв'язком. Цим і завершується доведення.

Чисельний метод розв'язування системи рівнянь, що далі розглядається, називається методом повного вилучення змінних або методом Жордана-Гауса.

Ідея методу полягає у багаторазовому переході від поточної системи до рівносильної з метою отримання такої системи, розв'язок якої визначається очевидним чином.

Метод являє собою ітераційну процедуру, на i-й ітерації розглядається i-те рівняння поточної системи. У i-му рівнянні визначається ненульовий коефіцієнт a'_{ij} при деякій змінній x_j. Мета

перетворення системи на і-й ітерації полягає в тому, щоб у рівносильній системі у і-тому рівнянні отримати при змінній x_j коефіцієнт, що дорівнює одиниці, а в усіх інших рівняннях при цій змінній отримати нульовий коефіцієнт (вилучити змінну x_j з усіх рівнянь, крім і-того). Визначений коефіцієнт a'_{ij} називають ведучим елементом. У матриці коефіцієнтів системи рівнянь рядок та стовпчик, що містять ведучий елемент, називають відповідно ведучим рядком та ведучим стовпчиком.

Для того, щоб отримати у ведучому рядку одиничний коефіцієнт при x_j, треба поділити ведучий рядок на ведучий елемент. Нові значення коефіцієнтів ведучого рядка визначаються формулою

$$a''_{is} = a'_{is} / a'_{ij}, \; s = \overline{1, n.} \quad (2.1)$$

Нові значення коефіцієнтів інших рядків визначаються умовою нульового нового значення коефіцієнта при змінній x_j. Для того, щоб виконати цю умову у довільному рядку r, треба перетворений ведучий рядок помножити на старе значення коефіцієнту у рядку r при x_j і результат відняти від цього рядка. Формула обчислення нового значення коефіцієнта має вигляд

$$a''_{rs} = a'_{rs} - a'_{is} a'_{rj} / a'_{ij}, \; s = \overline{1, n}; \; r \; i \neq \quad (4.4.4.2)$$

Аналогічно перетворюються і праві частини рівнянь

$$b''_i = b'_i / a'_{ij};$$

$$b''_r = b'_r - b'_i a'_{rj} / a'_{ij}, \; s = \overline{1, n} \; ; \; r \; \text{i.} \neq (4.4.4.3)$$

Перетворення матриці коефіцієнтів системи рівнянь за формулами (4.4.4.2) – (4.4.4.2) називається симплексним перетворенням.

Якщо при виконанні поточної ітерації виявляється, що рядок коефіцієнтів не містить жодного ненульового елемента, тобто відповідне рівняння має вигляд $0x_1 + 0x_2 + \ldots + 0x_n = b'_i$, то відрізняють два випадки.

1. $b'_i = 0$. У цьому випадку рівняння задовольняється при будь-якому наборі значень змінних $x_1, x_2, \dots, x_n$, і тому рівняння вилучається без втрати.

≠2. $b'_i \neq 0$. У цьому випадку рівняння не задовольняється при жодному наборі значень змінних $x_1, x_2, \dots, x_n$, і тому поточна система рівнянь не сумісна. Оскільки ця система рівносильна вихідній (на кожній ітерації відбувається перехід до рівносильної системи), то і вихідна система не сумісна.

Якщо система сумісна то у результаті застосування методу Жордана-Гауса утворюється система m^* рівнянь ($m^* \leq m$, де m - кількість рівнянь у вихідній системі). Кожне і-те рівняння має одну змінну x'_i, що вилучена з усіх інших рівнянь. Ці змінні називаються базисними, усі інші змінні – небазисними. Якщо множину індексів небазисних змінних позначити як N, то отриману систему можна уявити у вигляді

$$x'_i + \sum_{j \in N} a_{ij} x_j = b'_i,\ i = \overline{1, m^*}.$$

Розв'язавши отриману систему відносно базисних змінних, отримуємо загальний розв'язок системи.

$$x'_i = b'_i - \sum_{j \in N} a_{ij} x_j,\ i = \overline{1, m^*}.$$

Присвоюючи довільні значення небазисним змінним можна отримати відповідні значення базисних змінних Це дає змогу отримати усі сукупності значень змінних, що задовольняють системі рівнянь, тобто усі розв'язки системи. Саме тому систему і називають загальним розв'язком.

Розв'язок, що відповідає конкретним значенням змінних, називається частинним. Частинний розв'язок, у якого усі небазисні змінні дорівнюють нулю, називається базисним розв'язком.

Правило прямокутника

Перед тим, як розглянути приклад розв'язування системи рівнянь методом Жордана-Гауса, зупинимося на одному зручному способі обчислення значень коефіцієнтів системи при симплексному перетворенні. Перетворимо формулу (2.2)

$$a''_{rs} = a'_{rs} - a'_{is}\, a'_{rj} / a'_{ij} = \frac{a'_{rs}\, a'_{ij} - a'_{is}\, a'_{rj}}{a'_{ij}}.$$

Обчислення чисельника виразу (2.6) має наочне уявлення у матриці коефіцієнтів

$$\begin{pmatrix} a'_{11} \dots a'_{1j} \dots a'_{1s} \dots a'_{1n} \\ \cdot \quad \cdot \quad \cdot \\ a'_{i1} \dots a'_{ij} \dots a'_{is} \dots a'_{in} \\ \cdot \quad (+) \quad \cdot \quad (-) \quad \cdot \\ a'_{r1} \dots a'_{rj} \dots a'_{rs} \dots a'_{rn} \\ \cdot \quad \cdot \quad \cdot \\ a'_{m1} \dots a'_{mj} \dots a'_{ms} \dots a'_{mn} \end{pmatrix}$$

Ведучий елемент a'_{ij} і елемент a'_{rs}, значення якого треба перерахувати, утворюють діагональ прямокутника, у вершинах якого розташовані елементи, приймаючі участь у формулі симплексного перетворення. Для обчислення нового значення коефіцієнта треба взяти добуток ведучого елементу, на елемент, розташований разом з ведучим на спільній діагоналі, відняти добуток елементів, розташованих на іншій діагоналі, і результат поділити на ведучий елемент. Це і є правило прямокутника. Зрозуміло, що при обчисленні правих частин рівнянь також можна користуватися правилом прямокутника.

Приклад розв'язування системи

$$\begin{cases} 2x_1 + 3x_2 + 4x_3 = 5, \\ 4x_1 + 2x_2 - 3x_3 = 6. \end{cases}$$

Табл. 4.4.4.1

x_1	x_2	x_3	b	Σ
2	3	4	5	14
4	2	-3	6	9
1	3/2	2	5/2	7
0	**-4**	-11	-4	-19
1	0	**-17/8**	1	-1/8
0	1	11/4	1	19/4
-8/17	0	1	-8/17	1/17
22/17	1	0	39/17	78/17

Послідовність обчислень коефіцієнтів будемо записувати у таблиці 4.4.4.1, стовпці якої відповідають змінним і правим частинам рівнянь. Крім того, у таблиці використовується додатковий стовпець, елементи у якому дорівнюють сумі елементів інших стовпців для кожного рядка. Використовуючи правило прямокутника для поновлення елементів у правому стовпчику і перевіряючи суму елементів, можна контролювати правильність обчислень.

Перші два рядки містять коефіцієнти вихідної системи рівнянь. Ведучий елемент (виділений жирним шрифтом) на першій ітерації дорівнює 2. Після виконання першої ітерації змінна x_1 вилучається з другого рівняння. На другій ітерації ведучий елемент дорівнює -4, і змінна x_2 вилучається з першого рівняння. Після виконання другої ітерації отримуємо загальний розв'язок

$$\begin{cases} x_1 = 1 + 17/8\, x_3, \\ x_2 = 1 - 11/4\, x_3. \end{cases}$$

Відповідний базисний розв'язок - $x_1 = -1/8$, $x_2 = 19/4$, $x_3 = 0$.

Для отримання інших загальних і базисних розв'язків треба виконати додаткові симплексні перетворення. Наприклад, щоб зробити базисною змінною змінну x_3 замість x_1, треба виконати симплексне перетворення

відносно ведучого елемента -17/8, як показано у останніх двох рядках табл.. Отриманий загальний розв'язок -

$$\begin{cases} x_2 = 39/17 - 22/17\ x_1, \\ x_3 = -8/17 + 8/17\ x_1. \end{cases}$$

Відповідний базисний розв'язок - $x_1 = 0$, $x_2 = 39/17$, $x_3 = -8/17$. Аналогічно можна знайти розв'язок з небазисною змінною x_2.

4.4.5. Сімплекс метод для задачі лінійного програмування

Основу обґрунтування складає теорія опуклих множин. Опукла множина – це сукупність точок лінійного векторного простору, що задовольняє деякій умові.

Відрізком, що з'єднує точки a та b лінійного векторного простору, називається сукупність точок $P = \{ x \}$, що визначається таким чином: $x = (1 - \lambda)\ a + \lambda\ b$, де $\lambda \in [0, 1]$.

Точки a та b називаються кінцевими точками. Значенню $\lambda = 0$ відповідає точка a, і $\lambda = 1$ – точка b. При $0 < \lambda < 1$ x лежить у середині відрізку. Наведене означення відрізку повністю узгоджується із звичайним поняттям відрізку на площині. Як бачимо на мал.5.1, вектори x – a і b - a колінеарні. Тому $x - a = (b - a)\lambda$, і $x = (1-\lambda)a + \lambda b$, де $0 \le \lambda \le 1$.

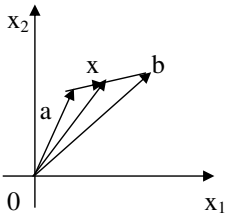
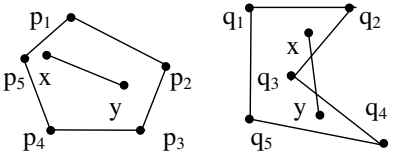


Рис.. 4.4.5.1 Геометрична ілюстрація визначення відрізку.

Опуклою множиною називається сукупність точок лінійного векторного простору, що разом з будь-якими двома точками a і b містить і відрізок [a, b], який їх з'єднує. На Рис.. 4.4.5.1 наведені приклади опуклої і неопуклої множин.

Крайньою точкою (вершиною або кутом) опуклої множини називається точка цієї множини, яка не лежить у середині жодного відрізку, що належить цій множині. На мал.5.2 p_i - крайні точки.



Мал.5.2. Приклади множин: (а) – опуклої і (б) – неопуклої.

Опуклою лінійною комбінацією точок $a_1, a_2, \dots, a_k$ називається лінійна комбінація $\sum_{j=1}^{k} \lambda_j a_j$, коефіцієнти якої задовольняють умові

$$\sum_{j=1}^{k} \lambda_j = 1 \text{ і } \lambda_j \geq 0, \; j = 1, k.$$

Сукупність усіх опуклих лінійних комбінацій точок $a_1, a_2, \dots, a_k$ називається опуклою оболонкою цих точок. Опукла оболонка є прикладом опуклої множини.

Відстанню $d(a, b)$ між двома точками $a = (a_1, a_2, \dots, a_n)$ та $b = (b_1, b_2, \dots, b_n)$ називають довжину вектору $a - b$: $d(a, b) = \sqrt{(a-b, a-b)} = \sqrt{\sum_{j=1}^{n} (a_j - b_j)^2}$.

Кулею S_c^r радіуса r з центром у точці c називається множина усіх точок x таких, що відстань $d(x, c) \le r$: $S_c^r = \{x \mid d(x, c) \le r\}$.

Множина точок називається обмеженою, якщо вона міститься у кулі скінченного радіусу.

Прикладом обмеженої множини є п'ятикутник на Рис.. 4.4.5.1, а прикладом необмеженої множини може бути півплощина.

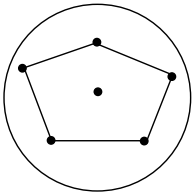


Рис.. 4.4.5.2. Приклад обмеженої множини.

Точка a множини M називається межевою, якщо кожна куля з центром у a містить як точки, що належать M, так і точки, що M не належать. Сукупність усіх межевих точок множини утворює її межу. Множина може містити свою межу чи ні. Наприклад, визначена вище куля містить свою межу, а множина $M = \{x \mid d(x, c) < p\}$, з тією ж межою, свою межу не містить.

Граничною точкою множини M називається границя послідовності точок множини M. Множина точок у n-вимірному лінійному векторному просторі називається замкненою, якщо вона містить усі свої граничні точки.

Можна показати, що кожна межева точка є граничною точкою, і тому замкнена множина завжди містить усі свої межеві точки, зокрема свою межу.

Множина може бути замкнена і необмежена або обмежена і незамкнена. Наприклад, на півплощина, точки якої задовольняють нерівності x_2 0, являє приклад необмеженої замкненої множини, а на півплощина, якій відповідає нерівність $x_2 > 0$, - необмеженої, незамкненої множини. Розглянута вище множина $M = \{x \mid d(x, c) < p\}$ обмежена і незамкнена.

Опукла множина може містити довільну кількість крайніх точок. На мал.5.2 наведено приклад опуклої множини із скінченою ненульовою кількістю крайніх точок. Коло (рис.. 4.4.5.2) має нескінчену кількість

крайніх точок – кожна точка межі. На півплощина не має жодної крайньої точки.

Опуклим многогранником називається обмежена замкнена опукла множина, що містить скінченну кількість крайніх точок. Приклад опуклого многогранника - многокутник на площині.

Властивості задачі лінійного програмування

Надалі будемо розглядати канонічну форму задачі:

$$F = (c,x) \rightarrow \min$$

$$Ax = b,$$

$$x \geq 0 .$$

Будемо вважати, що матриця A містить m рядків і n стовпців, усі рядки матриці A лінійно незалежні, тобто rank (A) = m.

Теорема 4.4.5.1 (Про опуклість допустимої області) Допустима область задачі лінійного програмування є опуклою множиною.

Доведення. Нехай u та v – довільні різні допустимі розв'язки задачі. Це означає, що мають місце такі співвідношення

$$Au = b, \quad Av = b, \quad u \geq 0, \quad v \geq 0.$$

Розглянемо довільну точку x на відрізку, що з'єднує точки u та v.

$$x = (1-\lambda)u + \lambda v, \quad 0 \leq \lambda \leq 1.$$

Покажемо, що ця точка задовольняє обмеженням задачі.

$$Ax = A((1-\lambda)u + \lambda v) = Au(1-\lambda) + Av\lambda = b(1-\lambda) + b\lambda = b.$$

$$u \geq 0, \ (1-\lambda) \geq 0 \Rightarrow (1-\lambda)u \geq 0.$$

$$v \geq 0, \lambda \geq 0 \Rightarrow \lambda v \geq 0.$$

$$x = (1-\lambda)u + \lambda v \geq 0.$$

Тому увесь відрізок, що з'єднує u та v, належить допустимій області, що і завершує доведення теореми.

Тепер з'ясуємо питання про кількість крайніх точок у допустимій області.

Теорема 4.4.5.2. (Про характерну властивість допустимого базисного розв'язку). Допустимий розв'язок задачі є крайньою точкою тоді і тільки тоді, коли він є базисним розв'язком.

Доведення. Спочатку доведемо достатність умови, тобто частину теореми "тоді". Нехай $x^0 = (x^0_1, x^0_2, \dots, x^0_n)$ допустимий базисний розв'язок. Позначимо через M множину індексів базисних змінних: $M = \{ j \mid x_j$ – базисна змінна $\}$. Як і раніше, множину індексів небазисних змінних будемо позначати через N. Будемо вважати, що B – під матриця матриці A, утворена базисними стовпцями. Позначимо x^0_B вектор, утворений базисними координатами вектору x^0, і x^0_N - вектор, утворений небазисними координатами вектору x^0_N. Для векторів x^0, x^0_B і x^0_N мають місце співвідношення

$$Ax^0 = b,\ x^0 \geq 0;\quad Bx^0_B = b;\ x^0_N = 0.$$

Доведення проведемо від супротивного. Припустимо, що x^0 не є крайньою точкою. Тоді існують два різні допустимі розв'язки $u = (u_1, u_2, \dots, u_n)$ та $v = (v_1, v_2, \dots, v_n)$, $u \neq v$, і x^0 лежить у середині відрізку $[u, v]$, тобто

$$x^0 = (1-\lambda)u + \lambda v,\ 0 < \lambda < 1.$$

З останнього векторного рівняння отримуємо систему лінійних рівнянь для небазисних координат векторів u та v

$$0 = (1-\lambda)u_j + \lambda v_j \ \forall j \in N. \quad (4.4.5.1)$$

Оскільки u та v допустимі розв'язки, $u_j \geq 0$ і $v_j \geq 0$. Крім того, $(1-\lambda) > 0$ і $\lambda > 0$. Тому система (4.4.5.1) має єдиний розв'язок $u_j = v_j \ \forall j \in N$.

Як допустимі розв'язки, u та v задовольняють рівнянням

$$Au = b,\ Av = b,$$

а враховуючи, що усі небазисні координати u та v дорівнюють нулю, отримуємо

$$B\bar{u} = \bar{b},\ B\bar{v} = \bar{b},$$

де $\bar{u}$, $\bar{v}$ утворені базисними координатами векторів u, v відповідно.

Отже x^0_B, $\bar{u}$ та $\bar{v}$ є розв'язками системи $Bx = b$, і оскільки B невироджена (система $Bx = b$ має тільки один розв'язок) $x^0_B = \bar{u} = v$. Тому $x^0 = u = v$, а це суперечить припущенню і завершує доведення достатності.

Тепер доведемо необхідність. Нехай $x^0 = (x^0_1, x^0_2, \dots, x^0_n)$ – крайня точка допустимої області. Знов доведення проведемо від супротивного. Припустимо, що x^0 не є допустимим базисним розв'язком. Позначимо через $N+$ множину індексів, для яких координати вектору x^0 мають додатні значення: $N+ = \{ j \mid x^0_j > 0 \}$. Тоді стовпці матриці A, номери яких належать $N+$, лінійно залежні, бо інакше x^0 – допустимий базисний розв'язок. Звідси прямує існування лінійної комбінації стовпців матриці A

$$\sum_{j \in N+} \lambda_j a^j = 0, \quad (4.4.5.4)$$

у якій не усі коефіцієнти дорівнюють нулю.

Побудуємо вектор $\lambda' = (\lambda'_1, \lambda'_2, \dots, \lambda'_n)$ таким чином

$$\lambda'_j = \lambda_j, \text{ якщо } j \in N+; \lambda'_j = 0, \text{ якщо } j \notin N+.$$

З (4.4.5.4) прямує $\sum_{j=1}^{n} \lambda'_j a^j = 0$, або $A\lambda' = 0$.

Якщо взяти число p, задовольняюче співвідношенню

$$0 < p < \min_{\forall \lambda'_j \neq 0} \{ x^0_j / | \lambda' | \},$$

то вектори $x^1 = x^0 + p\lambda'$, $x^2 = x^0 - p\lambda'$ задовольняють умовам

$$x^1 \geq 0,\ x^2 = \geq 0,\ Ax^1 = b,\ Ax^2 = b.$$

Дійсно, $x^1 = \geq 0$, оскільки, якщо $\lambda'_j \geq 0$, $x^1_j = x^0_j + p\lambda'_j \geq 0$, як сума невід'ємних величин, а при $\lambda'_j < 0$ $x^1_j = x^0_j - p|\lambda'_j| \geq . x^0_j - (x^0_j / |\lambda'_j|)|\lambda'_j| = 0$. Для x^2_j маємо. При $\lambda'_j \leq 0$ $x^2_j = x^0_j - p\lambda'_j \geq 0$, як сума невід'ємних величин, а при $\lambda'_j > 0$ $x^2_j = x^0_j - p|\lambda'_j| \geq . x^0_j - (x^0_j / |\lambda'_j|)|\lambda'_j| = 0$.

$$Ax^1 = A(x^0 + p\lambda') = Ax^0 + pA\lambda' = b + 0 = b.$$

Аналогічно, $Ax^2 = b$.

Таким чином x^1 і x^2 - допустимі розв'язки. Вектор x^0 можна уявити у вигляді $x^0 = 1/2\ x^1 + 1/2\ x^2$, це суперечить припущенню, що x^0 - крайня точка, і завершує доведення необхідності.

Остання теорема встановлює еквівалентність понять крайня точка допустимої області задачі лінійного програмування і допустимий базисний розв'язок задачі. Кількість допустимих базисних розв'язків скінчена, оскільки не перевищує кількості базисних розв'язків системи $Ax = b$, що дорівнює C^n_m. Надалі допустимий базисний розв'язок будемо називати опорним планом. Кожний опорний план являє вершину допустимої області, і, навпаки, кожна вершина – опорний план. З теорем 4.4.5.1і 4.4.5.2 прямує, що допустима область задачі лінійного програмування є опуклою множиною із скінченою кількістю крайніх точок.

Теорема 4.4.5.3 (Про уявлення обмеженої допустимої області задачі лінійного програмування). Якщо допустима область задачі лінійного програмування обмежена, то вона є опуклою оболонкою опорних планів.

Доведення. Для доведення теореми досить показати, що кожний допустимий розв'язок являє собою опуклу лінійну комбінацію опорних планів, оскільки з опуклості допустимої області задачі лінійного програмування прямує, що кожна опукла лінійна комбінація допустимих розв'язків є допустимим розв'язком.

Нехай $x^0 = (x^0_1, x^0_2, \ldots, x^0_n)$ – довільний допустимий розв'язок. Позначимо через N+ множину індексів додатніх координат вектору x^0.

Якщо стовпці матриці A з номерами, що належать N+, лінійно незалежні, то x^0 – опорний план, і твердження теореми виконується. Тому розглянемо випадок, коли вказані стовпці лінійно залежні, і виконується рівняння

$$\sum_{j=1}^{n} \lambda_j a^j = 0,$$

у якому не усі коефіцієнти дорівнюють нулю.

Як і при доведенні попередньої теореми, побудуємо вектор $\lambda' = (\lambda'_1, \lambda'_2, \dots, \lambda'_n)$ таким чином

$$\lambda'_j = \lambda_j, \text{ якщо } j \in N+; \lambda'_j = 0, \text{ якщо } j \notin N+.$$

$$\sum_{j=1}^{n} \lambda'_j a^j = 0, \text{ або } A\lambda' = 0.$$

Серед ненульових координат вектору λ' є як додатні, так і від'ємні, бо інакше, наприклад, при відсутності від'ємних, вектор $x^0 + t\lambda'$ при будь-якому значенні $t \geq 0$ належить допустимій області. А це суперечить обмеженості допустимої області, оскільки $| x^0 + t\lambda' |$ може приймати як завгодно великі значення при досить великому t.

Визначимо два числа p_1 і p_2 таким чином

$$p_1 = \min_j \{ x^0_j / (-\lambda'_j) \mid \lambda'_j < 0 \}; \quad p_2 = \min_j \{ x^0_j / \lambda'_j \mid \lambda'_j > 0 \}.$$

Тоді вектори $x^+ = x^0 + p_1\lambda'$ і $x^- = x^0 - p_2\lambda'$ належать допустимій області, що можна показати, як при доведенні теореми 5.2 для векторів x^1 і x^2.

Вектор можна уявити у вигляді опуклої лінійної комбінації

$$x^0 = p_2 / (p_1 + p_2) x^+ + p_1 / (p_1 + p_2) x^-,$$

де кожний з векторів x^+ і x^- має меншу кількість ненульових координат, ніж x^0. Зменшення кількості ненульових координат зумовлено тим, що для значень індексів, при яких відношення $\{ x^0_j / (-\lambda'_j) \}$ і x^0_j / λ'_j мають мінімальні значення, координати векторів x^+ і x^0 відповідно дорівнюють нулю.

Якщо ненульовим координатам кожного з векторів x^+ і x^- відповідають ленійно незалежні стовпчики матриці A, твердження теореми виконується. У іншому випадку процедуру розкладання векторів можна продовжити.

Оскільки матриця A не може містити нульових стовпчиків (інакше допустима область необмежена – відповідна координата може приймати як завгодно великі значення), кожний стовпчик являє множину лінійно незалежних векторів, і процес розкладання скінчений.

У результаті вектор x^0 уявляється у вигляді

$$x^0 = \sum_{j=1}^{k} \tilde{\lambda}_j x^j,$$

де $\sum_{j=1}^{k} \tilde{\lambda}_j = 1$, $\tilde{\lambda}_j \geq 0$, $j = 1, k$, x^j,- опорний план для усіх $j = 1, k$. Цим і завершується доведення.

4.5 . Основні принципи дослідження нелінійних моделей

4.5.1. Оцінка ефективності методів оптимізації

Велика різноманітність ітераційних алгоритмів ставить перед користувачем задачі вибору. Для цього слід визначити критерії, на основі яких один алгоритм буде вважатися більш доцільним, небажаним іншим. З цією метою зазвичай використовують такі оцінки:

1. *Точність пошуку* – значення околиці локального оптимуму, який призводить алгоритм після виконання заданого числа ітерацій.
2. *Швидкість збіжності* – число ітерацій, необхідне для досягнення заданої точності.
3. *Час рахунку* – час пошуку на ЕВМ локального оптимуму із заданою точністю, віднесене до коефіцієнта складності завдання (або до швидкої дії ЕВМ).

4. *Стабільність* – властивість алгоритму несуттєво збільшує число ітерацій при малих викликах вибору початкових точок, а також внаслідок погрішності обчислень.
5. *Надійність* - властивість алгоритму приведення до оптимального при багатократному повторенні пошуку з різних початкових точок.

Для порівняння алгоритмів за цими критеріями слід виробляти розрахунки в однакових або близьких умовах.

Класифікація методів на основі порядку похідних при виборі напрямків

1) методи нульового порядку – використовують лише значення функції;
2) методи першого порядку – використовують значення функцій і вектори перших продуктів;
3) методи другого порядку, в яких визначаються матриці другого виробництва;
4) методи більш високих порядків зазвичай не застосовуються.

Класифікація методів на основі вибору методу апроксимації цільової функції

1) лінійні методи, в яких використовується локальна апроксимація функцій в околицях точки лінійним поліномом за допомогою вектору – градієнта $F'(X)$;
2) квадратичні методи, в яких прийнята локальна квадратична апроксимація з $F'(X)$ і матриці другого похідних $F''(X)$;

Лінії рівня є корисним інструментом для розуміння поведінки функцій та їх залежностей від різних змінних. Вони дозволяють виявити зони з однаковими значеннями функції та визначити контури або області, де функція досягає певних значень.

Наприклад, у двовимірному просторі функцію f(x, y) можна відображати за допомогою ліній рівня. Кожен контур на графіку (Рис 2.1) представляє значення функції, яке залишається постійним на цьому контурі. Таким чином, можна відслідковувати зміну значень функції у різних частинах простору.

Лінії рівня широко використовуються в оптимізації та візуалізації функцій,

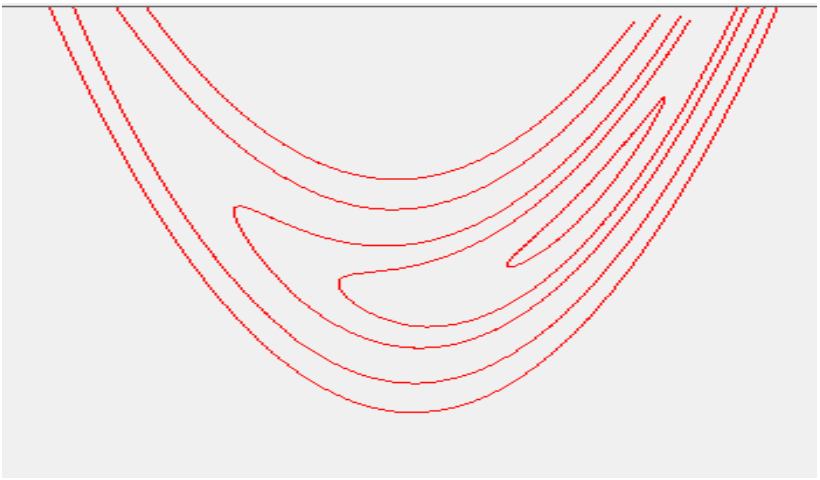


Рис 4.5.1 . Лінії рівня. На малюнку – функція Розенброка

Однією з важливих задач аналізу є завдання відшукання екстремуму (найбільшого або найменшого значення) скалярної функції *f(x)* *n*- мірного векторного аргументу *x* при деяких обмеженнях. Це завдання ми будемо записувати в такий спосіб можна формулювати так: знайти мінімум (або максимум) функції

$$f(\boldsymbol{x}), \boldsymbol{x} \in E^n, \qquad (4.5.1)$$

за виконання умов

$$fi(\boldsymbol{x})\ ОП_i\ 0,\ \ i=1,...,m, \qquad (4.5.2)$$

де $ОП_i$ одна з операцій порівняння $ОП_i \in \{ \leq, =, \geq \}$.

Умови (4.5.2) визначають деяку підмножину X *n*- вимірного евклідового простору E^n Надалі будемо називати X допустимою множиною задачі (4.5.1) - (4.5.2), а точки, що належать X, - її допустимими точками. Зауважимо, що завдання максимізації функції / (x) теж можна записати в вигляді (0.1) - (0.2), замінивши f (x) на f (x) = - / (x).

Задача безумовної мінімізації (без обмежень) функції багатьох змінних привернула увагу математиків за часів, коли закладалися основи математичного аналізу. Вона багато в чому стимулювала створення диференціального обчислення, а необхідна умова екстремуму (рівність градієнта нулю), отримане Ферма в 1629 р, стало одним з перших великих результатів аналізу. Пізніше в роботах Ньютона і Лейбніца були по суті сформульовані умови екстремуму II порядку (в термінах других похідних) для цього завдання. Сучасні чисельні методи аналізу задачі базуються на:

- алгоритмах безумовної мінімізації функції однієї змінної:

- алгоритмах знаходження безумовного екстремуму функції кількох змінних ($X \in E^n$) :

- завдання на відносний екстремум, або завдання мінімізації функції кількох змінних при наявності обмежень типу рівностей, коли *X* - множина рішень рівняння

g (X) = 0,

де *g (X)* є *m*-мірний вектор-функція, $m < n$.

У цьому розділі будуть послідовно розглянуті завдання безумовної мінімізації функції однієї змінної (X = £ г), завдання знаходження безумовного екстремуму функції кількох змінних (X = ЕЛ) і, нарешті, завдання на відносний екстремум, т. Е. Завдання мінімізації функції кількох змінних при наявності обмежень типу рівностей, коли X - безліч рішень рівняння

g (x) = 0,

де g (x) є m- мірний вектор-функція, m < n.

Багато спеціалістів вважають, що основним чисельним методом сучасної оптимізації є метод градієнтного спуску. У певному сенсі цей метод породжує більшість інших чисельних методів оптимізації. Метод градієнтного спуску активно використовується в обчислювальній математиці не тільки для безпосереднього вирішення завдань оптимізації (мінімізації), а й для завдань, які можуть бути переписані на мові оптимізації [1, 3, 6, 7, 8] (рішення нелінійних рівнянь, пошук рівноваги, обернені задачі і т.д.). Метод градієнтного спуску можна використовувати для завдань оптимізації в нескінченновимірних просторах [10], наприклад, для чисельного рішення задач оптимального управління [12, 13].

Методи дослідження функцій класичного аналізу є найбільш відомі методи вирішення нескладних оптимальних задач, які відомі з курсу математичного аналізу. Звичайною областю використання даних методів є завдання з відомим аналітичним виразом критерію оптимальності, що дозволяє знайти не дуже складне, також аналітичний вираз для похідних. Отримані при рівнянням нулю похідних рівняння, що визначають екстремальні рішення оптимальної завдання, вкрай рідко вдається вирішити аналітичним шляхом, тому, як, правило, застосовують обчислювальні машини. При цьому треба вирішити систему кінцевих рівнянь, найчастіше нелінійних, для чого доводиться використовувати чисельні методи, аналогічні методам нелінійного програмування. Додаткові труднощі при вирішенні оптимальної завдання методами дослідження функцій класичного аналізу виникають внаслідок того, що система рівнянь, що отримується в результаті їх застосування, забезпечує лише необхідні умови оптимальності. Тому всі рішення даної системи (а їх може бути і декілька) повинні бути перевірені на достатність. В результаті такої перевірки спочатку відкидають рішення, які не визначають екстремальні значення критерію оптимальності, а потім серед залишених

екстремальних рішень вибирають рішення, яке задовольняє умовам оптимальної завдання, т. Е. Найбільшому або найменшим значенням критерію оптимальності залежно від постановки задачі.

Методи дослідження при наявності обмежень на область зміни незалежних змінних можна використовувати тільки для відшукання екстремальних значень всередині зазначеної області. Особливо це відноситься до завдань з великим числом незалежних змінних (практично більше двох), в яких аналіз значень критерію оптимальності на кордоні допустимої області зміни змінних стає досить складним.

Метод множників Лагранжа застосовують для вирішення завдань такого ж класу складності, як і при використанні звичайних методів дослідження функцій, але при наявності обмежень типу рівностей на незалежні змінні. До вимоги можливості отримання аналітичних виразів для похідних від критерію оптимальності при цьому додається аналогічну вимогу щодо аналітичного виду рівнянь обмежень. В основному при використанні методу множників Лагранжа доводиться вирішувати ті ж завдання, що і без обмежень. Деяке ускладнення в даному випадку виникає лише від введення додаткових невизначених множників, внаслідок чого порядок системи рівнянь, розв'язуваної для знаходження екстремумів критерію оптимальності, відповідно підвищується на число обмежень. В іншому, процедура пошуку рішень і перевірки їх на оптимальність відповідає процедурі вирішення завдань без обмежень. Множники Лагранжа можна застосовувати для вирішення задач оптимізації об'єктів на основі рівнянь з приватними похідними і завдань динамічної оптимізації. При цьому замість вирішення системи кінцевих рівнянь для відшукання оптимуму необхідно інтегрувати систему диференціальних рівнянь.

Слід зазначити, що множники Лагранжа використовують також як допоміжний засіб і при вирішенні спеціальними методами завдань інших класів з обмеженнями типу рівностей, наприклад, у варіаційному численні

і динамічному програмуванні. Особливо ефективним є застосування множників Лагранжа в методі динамічного програмування, де з їх допомогою іноді вдається знизити розмірність розв'язуваної задачі.

Для широкого класу алгоритмів, що відповідають схемі (4.5.3) і є релаксаційними, важливими та принциповими є питання, що пов'язані з обґрунтованим вибором напрямків спуску та кроків, що використовують алгоритми цієї схеми. В усіх цих випадках питання пов'язані з порівняльним аналізом і обґрунтуванням найбільш поширених технологій вибору напрямків та кроків які використовують алгоритми, є принциповими і важливим. Методи на кожній ітерації виконують наступні обчислення

$$x_{k+1} = x_k - h_k\, v'(x_k),\ k = 0, 1. \ldots \qquad f(x_k) > f(x_{k+1}), \qquad (4.5.3)$$

і проблема вибору параметрів $- h_k\, v'(x_k)$, є важливою в методах нелінійного програмування, не має теоретичного розв'язку [2] і потребує експериментального дослідження. Звичайно теоретичну якість алгоритмів, що відповідають наведеній схемі характеризують параметром, що визначається залежністю:

$$\| x^{j+1} - x^* \| <= g_j \| x^j - x^* \| \quad \text{або} \quad \| x^{j+1} - x^* \| <= g \| x_j - x^* \|^2 \quad (4.5.4)$$

Залежність і параметр g визначає швидкість збіжності алгоритмів. Теоретична оцінка якості алгоритмів, на практиці, не завжди збігається з практичними результатами і це повинна підтвердити програма, що пропонується. На практиці ефективність алгоритмі залежить як від вигляду функції, що мінімізується так і від характеристик алгоритму – технології вибору напрямку спуску та кроку алгоритму. Наприклад, для простої функції $f(x,y) = x^2 + y^2$ необхідна лише одна ітерація алгоритму градієнтного спуску із знаходженням мінімуму функції вздовж напрямку градієнту, а для подібної простішої функції $f(x,y) = x^2 + 5y^2$ кількість

ітерацій (в залежності від необхідної точності) наближається до нескінченності.

Задачу математичного програмування, що досліджується, можна формулювати так:

$$f(x) \rightarrow \min \qquad (4.5.5)$$

де f: $R_m \rightarrow R$. Точка $x^* R_m$ називається рішенням завдання (або точкою глобального безумовного мінімуму функції f, якщо

$$f(x^*) \leq f(x) \qquad (4.5.6)$$

при всіх $x R_m$. Або Знайти $\boldsymbol{x}^*$, що доставляє мінімум (або максимум) функції $f(\boldsymbol{x})$, $\boldsymbol{x} \in En$. $\boldsymbol{x}^*$ визначається як гранична точка послідовності x_k, k =0, 1 ,2…

Якщо нерівність (2) виконано лише для х, що лежать в деякій околиці Vx^* точки x^*, то точка х * називається локальним рішенням задачі, або точкою локального безумовного мінімуму функції f. Якщо нерівність (4.5.6) виконується при всіх $x \neq x^*$, то говорять про строгий глобальний і, відповідно, строгий локальний мінімум. Рішення задачі (4.5.5) іноді позначають *argmin f (x)* (або, більш повно, *argminx Rm f (x)*; коли мова йде про завдання безумовної оптимізації в позначеннях *argminx Rm f (x) і minx Rm f (x)* опускають індекс "*x* з *Rm*"). Зазвичай з контексту ясно про який мінімум локальний, глобальний і т. д. йдеться.

Аналогічні поняття (максимумів) визначаються для задачі

$$f(x) \rightarrow \max.$$

Різняться алгоритми, що працюють за подібною схемою, стратегіями вибору кроку h_k та напрямку $v'(x_k)$.

Якщо

$$f(x_k) > f(x_{k+1}),$$

то алгоритм має назву релаксаційного. Для цього класу алгоритмів стратегії вибору кроку повинні задовольняти нерівностям

$\alpha \times f'(x_k), x_k - x_{k+1}) \leq f(x_k) - f(x_{k+1}),$

$\beta \times f'(x_k), x_k - x_{k+1}) \geq f(x_k) - f(x_{k+1}),$

де $\alpha, \beta, 0<\alpha<\beta<1$, — деякі числа (фіксовані параметри).

Зупиняється алгоритм в точці локального екстремуму. Точка x_0 називається точкою локального максимуму функції $f(x)$, Якщо існує така околиця цієї точки, що для всіх x з цієї околиці виконується нерівність: $f(x) \leq f(x_0)$.

Ця стратегія має таку важливу перевагу перед не релаксаційними алгоритмами. Якщо функція *f (x)* обмежена знизу в *Rn*, то послідовність $\{f(x_k)\}$ $k = 0 \ldots \infty$ сходиться. У будь-якому випадку ми покращуємо початкове значення цільової функції.

Теорія показує, а практика розрахунків підтверджує, що метод найкрутішого спуску не дуже ефективний у випадках, коли криві рівнів цільової функції сильно розтягнуті, тобто є глибокі яри при пошуку мінімуму або діапазони при пошуку максимуму. Найкрутіший напрямок спуску майже ортогональний найкращому напрямку пошуку, як наслідок, оптимальний крок весь час зменшується, і алгоритм «зависає», не досягаючи екстремуму. Виходом із цієї ситуації може стати масштабування змінних, при якому лінії рівня наближаються до кола.

Щоб зменшити обсяг обчислень цільової функції, пов'язаних з чисельним визначенням частинних похідних, іноді використовують метод координатного спуску, який ще називають релаксаційним або методом Гаусса-Зейделя. Нехай e_i – вісь x_i одиничний вектор, а $x=\{x_1 \ldots, x_n$ – початкова точка пошуку. Один інтернаціонал координатного спуску повинен зробити кроки: $x_{k-1} = x_k + \lambda_k e_k$, $k = 1,\ldots, n$.

Крок, як і в методі найкрутішого спуску, визначається умовою max $f(x_k + \lambda_k s_k)$ Метод Гаусса-Зейделя страждав тим же недоліком, що і метод найкрутішого спуску, – погана збіжність за наявності ярів.

Один із способів подолання обчислювальних труднощів, пов'язаних зі структурою ярка цільової функції, передбачає використання інформації не тільки про її першу похідну, але й вищого порядку, що міститься в других часткових похідних.

Окремим випадком задачі нелінійної оптимізації програмування є задача квадратичного програмування, в якій обмеження є лінійними, а функція є сумою лінійної та квадратичної функції (квадратичної форми).

$$f(\boldsymbol{x}) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_i x_j \to \min$$

Як і загальному випадку розв'язання задач нелінійного програмування, визначення глобального екстремуму завдання квадратичного програмування немає ефективного обчислювального методу, якщо не відомо, що будь-який локальний екстремум є одночасно і глобальним. Оскільки завдання квадратичного програмування безліч допустимих рішень опукло, то, якщо цільова функція увігнута, будь-який локальний максимум є глобальним; якщо ж цільова функція – опукла, то будь-який локальний мінімум також і глобальний.

Функція f є сумою лінійної функції (яка є і опуклою, і увігнутою) і квадратичної форми. Якщо остання є увігнутою (опуклою), то завдання відшукання максимуму (мінімуму) цільової функції можуть бути успішно вирішені. Питання про те, чи буде квадратична форма увігнутою або опуклою, залежить від того, чи є вона негативно-визначеною, позитивно-визначеною, позитивно – напив визначеною або взагалі невизначеною.

Зазвичай передбачається, що $f(\boldsymbol{x})$ є диференційованою і опуклою функцією. Це дозволяє дати певні гарантії успіху застосування градієнтного спуску. На практиці градієнтний спуск успішно застосовується навіть тоді, коли проблема не має жодної з перерахованих

вище властивостей (приклад далі). Для розв'язання задачі пошуку *argmin* $f(\boldsymbol{x}), \boldsymbol{x} \in E^n$ існує альтернатива.

Ще в 17 столітті П'єром Ферма був придуманий критерій, який дозволяв вирішувати прості завдання оптимізації, а саме, якщо $\boldsymbol{x}^*$ точка мінімуму $f(\boldsymbol{x})$, то.

$\nabla f(\boldsymbol{x}^*)=0$

Цей критерій базується на лінійному наближенні

$$f(x) \approx f(x^*) + f'(x^*)(x - x^*).$$

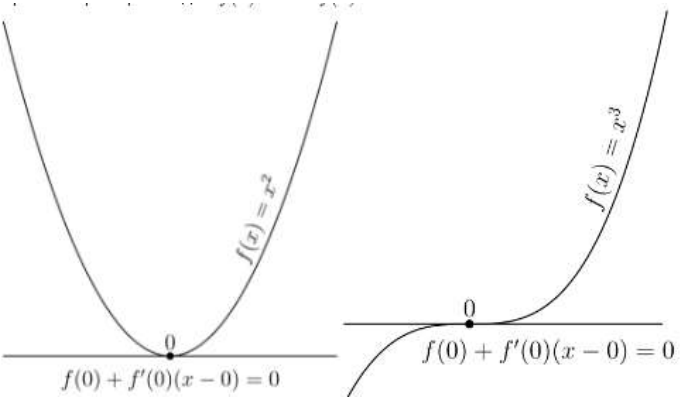


Рис 4.5.1

Чим ближче до $\boldsymbol{x}^*$ тим точніше це наближення. Також рівність градієнта нулю означає рівність всіх приватних похідних нулю, тому в багатовимірному випадку можна отримати цей критерій просто послідовно застосувавши одновимірний критерій щодо кожної змінної окремо. Надалі ми будемо порівнювати алгоритми за ефективністю.

Частинні похідні $f(\boldsymbol{x})$ в точці $\boldsymbol{x0}$ визначаються як межі (якщо існують)

$$\frac{\partial f(\boldsymbol{x}^0)}{\partial x_j} = \lim_{h \to 0} \frac{f(\boldsymbol{x}^0 + h\,\boldsymbol{e}_j) - f(\boldsymbol{x}^0)}{h},$$

де $ej = (0,...,0,1(1$ стоїть на j-у місці$),0,...,0)^T$ – *n-вимірний* одиничний вектор, що був транспонований у вектор-рядок.

Функція $f(x)$має назву *диференційованої* в точці x, якщо вона визначена в деякому околі точки x і для x із цього околу $f(x)$ можна подати у вигляді

$$f(x) = f(x0) + \nabla mf(x0)\,(x - x0) + o(||x - x0||).$$

Вектор

$$\nabla f(x^0) = \left(\frac{\partial f(x^0)}{\partial x_1}, \frac{\partial f(x^0)}{\partial x_2}, ..., \frac{\partial f(x^0)}{\partial x_n} \right)^T$$

градієнт функції.

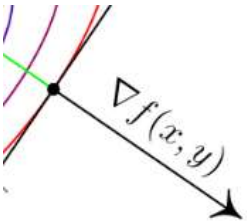


Рис 4.5.2

4.5.2. Загальні принципи побудови методів безумовної оптимізації

Як міру складності алгоритму найчастіше розглядають дві характеристики: час його роботи та обсяг використовуваної пам'яті, які виражаються як функції від розміру задачі. Оскільки час роботи алгоритму, як правило, не менше обсягу пам'яті, то надалі під складністю алгоритму, головним чином, ми розумітимемо його тимчасову складність (час роботи). Обчислювальні алгоритми, які ми вивчатимемо, вирішують деяке завдання, виконуючи певну послідовність елементарних арифметичних та логічних операцій (складань, віднімань, творів, поділів та

порівнянь). Тимчасова складність таких алгоритмів визначається як кількість елементарних операцій, що він виконує. Такий підхід до оцінки складності алгоритмів називають алгебраїчним. Алгебраїчний підхід ігнорує дискретність даних у пам'яті комп'ютера (там немає дійсних чисел, лише раціональні). У пам'яті комп'ютера під запис числа приділяється фіксована кількість бітів. Це обмежує розміри чисел, над якими арифметичні операції виконуються точно (без заокруглень) та з однаковою швидкістю. Якщо ж розміри чисел, над якими виконуються арифметичні операції, або результати цих операцій перевершують розміри комп'ютерної розрядної сітки, то для точного виконання арифметичних операцій потрібно використовувати бібліотеку алгоритмів для виконання арифметичних операцій з довгими (великими) числами

Використовуючи градієнтні методи, можна знайти рішення будь-якого завдання нелінійного програмування. Застосування цих методів у випадку дозволяє знайти точку локального екстремуму. Тому доцільно використовувати їх знаходження вирішення завдань опуклого програмування. Процес знаходження вирішення задачі за допомогою градієнтних методів полягає в тому, що починаючи з деякої точки здійснюється послідовний перехід до деяких інших точок доти, доки не буде знайдено прийнятне рішення вихідної задачі – точка, що задовольняє умовам зупинки обчислень. Градієнтні методи можуть бути поділені на дві групи.

До першої групи належать методи, при використанні яких досліджувані точки не виходять за межі області допустимих розв'язків задачі. В даному випадку найпоширенішим є метод Франка – Вульфа. До другої - методи, при використанні яких досліджувані точки можуть належати, так і не належати області допустимих рішень. Проте в результаті реалізації ітераційного процесу знаходиться точка області допустимих рішень, що визначає прийнятне рішення. Найчастіше використовуються

метод штрафних функцій та метод Ерроу – Гурвіца. При знаходженні рішення задачі градієнтними методами ітераційний процес триває доти, поки градієнт функції в черговій точці не стане рівним нулю або поки не виконається нерівність

Окрема технологія для випадку коли обмеження містять лише лінійні нерівності. Ця особливість є основою для заміни в околиці досліджуваної точки нелінійної цільової функції лінійної, у результаті рішення вихідної задачі зводиться до послідовного вирішення задач лінійного програмування.

Процес знаходження рішення починають із визначення точки, що належить області допустимих рішень. Обчислюють у цій точці градієнт функції :

$$\nabla f\left(X^{(k)}\right)=\left(\frac{\partial f\left(X^{(k)}\right)}{\partial x_1};\frac{\partial f\left(X^{(k)}\right)}{\partial x_2};\ldots;\frac{\partial f\left(X^{(k)}\right)}{\partial x_n}\right),$$

Та будують лінійну функцію, що базується на лінійному наближенні

$$F_k=\frac{\partial f\left(X^{(k)}\right)}{\partial x_1}x_1+\frac{\partial f\left(X^{(k)}\right)}{\partial x_2}x_2+\ldots+\frac{\partial f\left(X^{(k)}\right)}{\partial x_n}x_n.$$

Знаходять максимум функції при обмеженнях. Нехай вирішення цього завдання визначається точкою $X^{(k)}$. За нове допустиме вирішення $Z^{(k)}$ вихідного завдання приймають координати точки , які знаходять за формулами

$$X^{(k+1)}=X^{(k)}+\lambda_k\left(Z^{(k)}-X^{(k)}\right),$$

Замість того , щоб вирішувати завдання (1)-(3), знаходять максимум функції

$$F\left(x_1,x_2,\ldots,x_n\right)=f\left(x_1,x_2,\ldots,x_n\right)+H\left(x_1,x_2,\ldots,x_n\right),$$

де визначається системою обмежень і називається штрафною функцією Останню можна побудувати у різний спосіб. Найчастіше вона має вигляд.

Використовуючи штрафну функцію, послідовно переходять від однієї точки до іншої, доки не отримають прийнятне рішення.

Модифікації градієнтного спуску використовують співвідношення

$x_{k+1} = x_k - step_{k1}\, v_{k1} - step_{k2}\, v_{k2}$, $k = 0, 1 \ldots$ $x_{k+1}, x_k \in E^n$, $step_{k1}, step_{k2} \in E^1$

в якому напрямки v_{k1}, v_{k2} будують з допомогою градієнту.

Останнє доданок характеризує цю «інерційність», алгоритм щокроку намагається рухатися проти градієнта, але за цьому по інерції частково рухається у тому напрямі, як і попередньої ітерації. Такі методи мають дві важливі властивості: Вони мало ускладнюють простий градієнтний спуск в обчислювальному плані. При акуратному підборі такі методи набагато швидше, ніж звичайний градієнтний спуск навіть з оптимально підібраним кроком. Вони мало ускладнюють простий градієнтний спуск в обчислювальному плані. При акуратному підборі такі методи набагато швидше, ніж звичайний градієнтний спуск навіть з оптимально підібраним кроком.

Скалярні - множники перед напрямками v_{k1}, v_{k2}, що будують з допомогою градієнту називають довжиною кроку. Очевидно, вони повинні бути визначеними як позитивними. Існує багато модифікацій цих методів, які відрізняються один від одного стратегією обчислення довжини кроку на кожній (к-тій) ітерації.

Найбільш важливі з них наведені нижче.

1.Постійний крок. $step_{k1}\, step_{k2} = h > 0$, Для таких функцій, градієнт яких задовольняє умові Ліпшиця це можливо. Умова Ліпшиця - існує таке число L що нерівність

$$\|f'(x) - f'(y)\| < L\|x - y\|$$

має місце для всіх $f'(x), f'(y), x, y \in E^n$. В цьому випадку найкращим є максимальний крок $h = 1/L$[1].

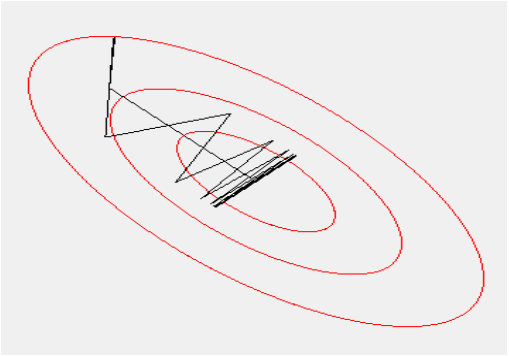


Рис. 4.5.2.1. Постійний крок з максимальним коефіцієнтом релаксації.

2. З дробленням кроку на кожній ітерації. Крок, на кожній ітерації, зменшується с заданим постійним коефіцієнтом якщо не досягається зменшення функції за формулою $x_{k+1} = x_k - step_{k1}\, v_{k1}$

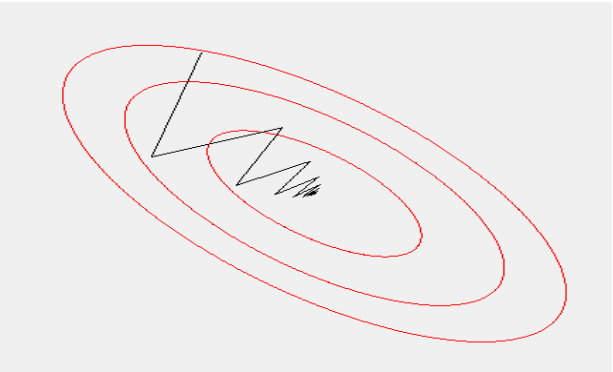


Рис. 4.5.2.2. З дробленням кроку на кожній ітерації. (повернена еліптична функція)

3. Найшвидший спуск (або метод повної релаксації)

h_k =arg min $x_k - step_{k1}\, v_{k1}$, мінімум визначається по $- step_{k1} > 0$.

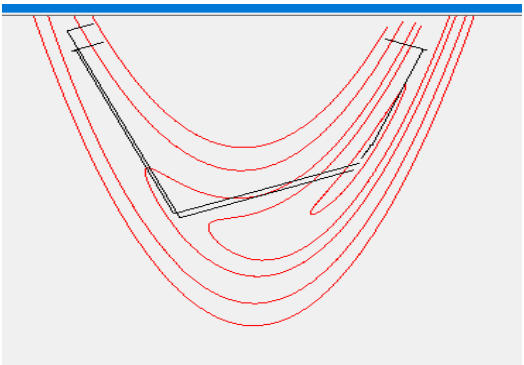


Рис. 4.5.2.3. Найшвидший спуск. Функція Розенброка.

4. З регулюванням кроку. Крок, на кожній ітерації, с заданим постійним коефіцієнтом якщо досягається зменшення функції за формулою $x_{k+1} = x_k - step_{k1}\, v_{k1}$

5. Рандомізований крок. Параметри $step_{k1}$ $step_{k2}$ обираються випадково

Збіжність послідовності

$x_1, \ldots, x_k, x_{k+1} \ldots \in E^n$,

гарантується якщо крок знаходиться відповідно до правила Голдштейна - Армійо: h такий, що мають місце нерівності

$x_{k+1} = x_k - hf'(x_k)$,

$\alpha^*(f'(x_k), x_k - x_{k+1}) \leq f(x_k) - f(x_{k+1})$,

$\beta^*(f'(x_k), x_k - x_{k+1}) \geq f(x_k) - f(x_{k+1})$,

де задані наперед α, β, $0<\alpha<\beta<1$, — деякі фіксовані параметри.

В останньому випадку якість алгоритму суттєво залежить від вибору фіксованих значень α, β.

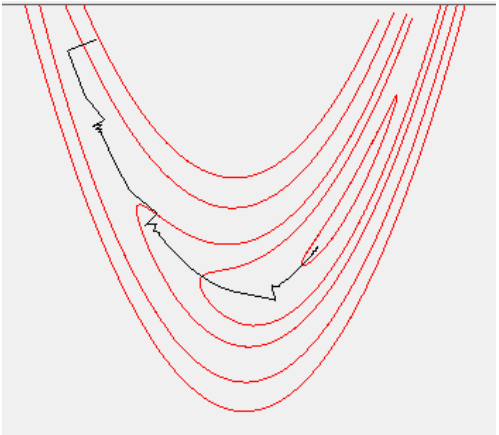


Рис. 4.5.2.3. Рандомізований крок

Метод сполучених градієнтів. Цей спосіб пошуку абсолютного екстремуму поєднує у собі поняття градієнта цільової функції та пов'язаних напрямів.

Визначення сполученості сформулюється наступним чином: два вектори x і y називають A - сполученими (або сполученими по відношенню до матриці A) або A-ортогональними, якщо скалярний добуток x і A * y дорівнює нулю, тобто: $x^T Ay=0$. Сполученість можна вважати узагальненням поняття ортогональності. Дійсно, коли матриця A – одинична матриця, відповідно до рівності 4, вектори x та y – ортогональні. Яким чином обчислювати пов'язані напрямки. Один із можливих способів обчислювати пов'язані напрямки – використовувати методи лінійної алгебри, зокрема, процес ортогоналізації Грама-Шмідта. Але для цього необхідно знати матрицю A, тому для більшості завдань (наприклад, навчання багатошарових нейромереж) цей метод годиться. Існують інші, ітеративні способи обчислення сполученого напрямку, найвідоміший – формула Флетчера-Рівса. Відповідно до методу Флетчера-Рівса напрямки спуску $C_к$ на кожній ітерації обчислюються за формулами:

$C_К = -f'(X_к) + в_{к-1} C_{к-1}$, $к > 1$,

$C_0 = -f'(X_0)$

Метод Флетчера - вважається одним із ефективних методів вирішення завдань великої розмірності. Має властивості збіжності ньютонівських і квазіньютонівських алгоритмів для сильно опуклих функцій, що тричі диференціюються, але не вимагає обчислення матриці других похідних і вирішення систем рівнянь на кожній ітерації. Мінімізує квадратичну функцію за n кроків. Зручний для завдань із слабко заповненими матрицями.

Метод Флетчера - Рівса побудований відповідно до загальної схеми методів спуску

$x_{k+1} = x_k - step_{k1}\, v_{k1} -- step_{k2}\, v_{k2},$

і використовує для одновимірного пошуку послідовність напрямків, кожен з яких є лінійною комбінацією антиградієнта в поточній точці та напрямку попереднього спуску. Напрямки кожної пари ітерацій є пов'язаними для модельної квадратичної функції з тими самими градієнтами. Найважливішою важливою особливістю алгоритму є необхідність мінімізації функції мети вздовж спуску на кожній ітерації.

Для обґрунтування нагадаємо загальну схему методів спуску

$x_{k+1} = x_k - step_{k1}\, v_{k1}$

де напрямок спуску утворює гострий кут з градієнтом у точці $X_{к,}$ або v_{k1} * grad<0.

Якщо спуск найшвидший, то добуток спуску на градієнт у точці *Хк,* дорівнює 0. Ці два напрямки є ортогональними.

Відповідно до методу Флетчера-Рівса напрямки спуску *Ск* на кожній ітерації обчислюються за формулами

$C_к = -f'(X_к) + в_{к-1}\, C_{к-1}, \; к > 1,$

$C_0 = -f'(X_0)$

Величини $в_{к-1}$ вибираються так щоб напрямки $С_к$, $С_{к-1}$ були $f''(Хк)$ - сполученими і, отже, мало місце співвідношення:

$Ск\, f''(X_к)\, С_{к-1} = 0 = (-f'(X_к))^t\, f''(Хк)\, С_{k-1} + в_{к-1}\, С_{к-1}\, f''(Хк)\, С_{к-1.}$

З останнього рівняння

$в_{к-1} = (-f'(X_к))^t\, f''(Хк)\, С_{k-1} / С_{к-1}\, f''(X_к)\, С_{к-1.}$

Точка $Хк+1$ визначається внаслідок мінімізації функції $f(X_к+r_кС_к)$ по r .Обчислення за формулою можна суттєво спростити, якщо для матриці других похідних скористатися деякою її апроксимацією.

Розглянемо розкладання градієнта в ряд Тейлора на околиці точки $Хк$ за ступенями компонент вектора $r_кС_к$

$f'(Хк-r_{к-1}С_{к-1}) = f'(Хк) - f''(Хк)(r_{к-1}Ск-1)$

Якщо функція f (X) є квадратичною, розкладання функції містить лише два члени, які наведені в останньому рівнянні. Для функцій загального виду остання формула є моделлю, точність якої зростає в міру наближення до точки екстремуму. З останньої формули

$f''(Хк)\, Ск-1 = (f'(Хк) - f'(Хк- r_{к-1}\, Ск-1))/ r_{к-1} = (f'(Хк) - f'(Хк-1))/ r_{к-1}$

Якщо підставити значення $f''(Хк)\, Ск-1$ з останньої формули рівняння, то отримаємо просту формулу для обчислення $в_{к-1}$ не містить матриць других похідних

$в_{к-1} = (f'(Хк))^t(f'(Хк)-f'(Хк-1))/ Ск-1\,(f'(Хк)-f'(Хк-1)).$

Коефіцієнти $r_{к-1}$ скорочуються. Також можна показати, що з ортогональності векторів $С_{к-1}$ і $f'(X_к)$ (ця ортогональність є наслідком того, що Хк отримана в результаті мінімізації функції вздовж напрямку $С_{к-1}$), після розкриття дужок та відкидання творів, які дорівнюють нулю можна отримати

$в_{к-1} = -(f'(Хк))^t f'(Хк)/Ск-1 f'(Хк-1).$

Це рівняння можна спростити і зменшити пам'ять необхідну для обчислень

Якщо рівняння переписати для $Ск-1$

Ск-1 = -f' (Хк-1) + вк-2 Ск-2,

помножити ліворуч і праворуч на *f' (Хк-1),* то, після відбиття членів рівних нулю (ортогональні вектори Ск-1 і -f' (Хк-1)) отримаємо

Ск-1 f'(Хк-1) = -f'(Хк-1) f'(Хк-1).

З цього рівняння для *Ck* подібним чином можна довести ортогональність градієнтів - *Ск f'(Хк-1) = -f'(Хк) f'(Хк-1).* Ця рівність можлива, якщо *Ск = - f '(Хк-1),* що не вірно, або якщо всі вектори попарно ортогональні. Отже *f '(Хк) f'(Хк-1) = 0.*

Остаточний варіант

$в_{к-1} = (f'(Хк))tf'(Хк)/(f'(Хк-1) f'(Хк-1).$

Тепер вид процедури обчислень наступний

1. На першій ітерації

$C_1 = - grad (X)$

2 На к -му кроці розв'язується задача одномірної мінімізації по r функції $f(x+ r C_к)$. І обчислюємо похідну у новій точці. Якщо похідна близька 0, то отримано розв'язання задачі

інакше визначаємо

Ск+1 = - f' (Хк) + Ск ((f' (Хк+1))^t f' (Хк+1)/ (f' (Хк) f' (Хк1)).

І повторимо обчислення.

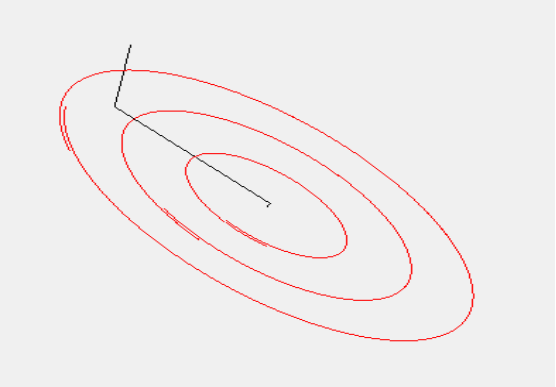


Рис 4.5.2.3. Метод сполучених градієнтів

4.5.3. Методи одновимірної оптимізації

Для вибору кроку методів нелінійної оптимізації схеми (4.5.3) використовуються методи одновимірного пошуку. Найбільш важливим є метод найшвидшого спуску (або метод повної релаксації). Крок обирається за умов $h_k = \arg\min f(x_k - hf'(x_k))$, мінімум визначається по $h>0$.

Цільові функції, які досліджуються є унімодальними, тобто мають на інтервалі дослідження, який розглядається, тільки один оптимум. Таке обмеження на характер цільової функції не є таким жорстким, як може здатися, так як багато задач, з якими інженер стикається в своїй практиці, виявляються унімодальними.

Чисельні методи, які орієнтуються на розв'язання задач безумовної оптимізації, можна розділити на три класи:

- *методи прямого пошуку*, що базуються на обчисленні тільки значень цільової функції;

Задача одновимірної оптимізації ставиться таким чином: значення параметра X цільової функції f(x), який називають проектним параметром, знаходяться на інтервалі дослідження [a, b]. В процесі пошуку оптимуму цільової функції цей інтервал, який називається інтервалом невизначеності, постійно зменшується (звужується), тому методи

одновимірної оптимізації іноді називають методами звуження інтервалу невизначеності.

В процесі одновимірної оптимізації цільової функції можна виділити два етапи:

1) встановлення меж відрізка, на якому реалізується процедура пошуку оптимуму;

2) зменшення відрізка до заданої похибки обчислення ε .

Перший етап реалізується за допомогою евристичних методів пошуку і є дуже складним. Другий - називають правилом виключення відрізків, реалізують алгоритм пошуку, що дозволяє знайти точку оптимуму шляхом послідовного виключення частин початкового обмеженого відрізка [a, b], тобто за допомогою ітераційних алгоритмів. В якості умови закінчення ітераційного процесу використовується момент, коли підінтервал, що залишився, зменшиться до достатньо малих розмірів (зазвичай для цього задають значення заданої похибки обчислення ε).

Очевидно, найбільш природнім способом звуження інтервалу невизначеності для одновимірної унімодальної функції є ділення його на декілька рівних частин з наступним обчисленням значень цільової функції в вузлах отриманої сітки (рис. 2).

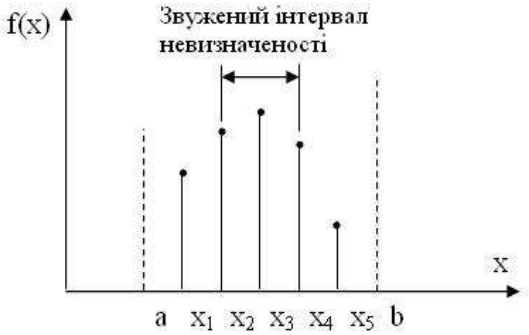


Рис 4.5.3.1 – Метод загального пошуку

В результаті інтервал невизначеності звужується до двох кроків сітки. Звичайно говорять про дроблення інтервалу невизначеності, яке характеризується коефіцієнтом f. Розділивши інтервал невизначеності на N частин, отримаємо N+1 вузол, і тоді

$$f = \frac{2}{N+1}.$$

Щоб отримати значення $f = 0,01$, необхідно обчислити цільову функцію в 199 точках, а при $f = 0,001$ $N=1999$. Звідси видно, що ефективність цього методу при зменшенні інтервалу невизначеності швидко падає. Напрошується інший варіант: щоб отримати $f=0,01$, необхідно обчислити спочатку функцію в 19 точках і отримати $f = 0,1$, а потім обчислити ще 19 значень функції на скороченому інтервалі невизначеності, отримати $f = 0,01$, зробивши при цьому всього 38, а не 199 обчислень. Таким чином, при деякій винахідливості ефективність пошуку можна різко збільшити.

Метод половинного ділення (розділення відрізка навпіл)**.** Суть метода полягає в постійному діленні відрізка дослідження цільової функції [a, b] навпіл і визначенні на ньому координат трьох точок x_1, x_2, x_m. При чому значення їх визначаються як:

$x_m = \frac{a+b}{2}$, L=b-a, $x_1 = a + \frac{L}{4}$, $x_2 = b - \frac{L}{4}$.

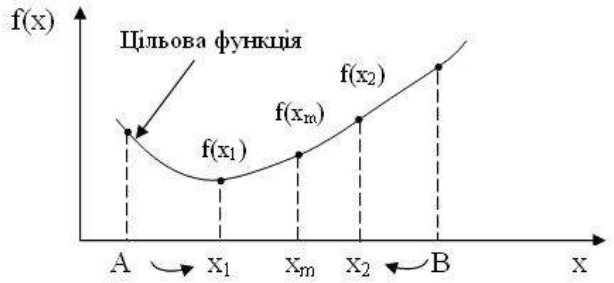


Рис 4.5.3.2 - Геометрична інтерпретація методу половинного ділення

Точки x_1, x_m, x_2, поділяють відрізок [a,b] на чотири рівні частини (рис. 3), обчислюємо значення цільової функції $f(x_1), f(x_2)$. Потім порівнюємо значення $f(x_1)$ і $f(x_m)$, якщо $f(x_1) < f(x_m)$, то виключаємо з дослідження відрізок $[x_m, b]$ та покладемо $b = x_m$. Тоді середньою точкою нового відрізка [a, b] стає $x_1(x_m = x_1)$. Але якщо $f(x_1) \geq f(x_m)$, то порівнюємо значення цільової функції $f(x_2)$ і $f(x_m)$; якщо $f(x_2) < f(x_m)$, то виключаємо відрізок $[a, x_m]$, покладемо $a = x_m$, $x_m = x_2$; якщо $f(x_2) \geq f(x_m)$, то виключаємо відрізок $[a, x_1]$ та $[x_2, b]$, покладемо $a = x_1$, $b = x_2$, тобто формуємо новий відрізок дослідження. Обчислюємо $L = b - a$, якщо $|L| < \varepsilon$, якщо немає, то знову повертаємося до початку.

Даний алгоритм , як і всі інші, ітераційний, тому зазвичай в якості умови закінчення ітераційного процесу обирають умову $|a - b| < \varepsilon$, тобто звуження відрізку виконується до тих пір, поки його величина не зменшиться до заданої обчислювальної похибки ε.

Метод “золотого перетину”

З кожних трьох значень цільової функції, які були обчислені в інтервалі невизначеності в подальшому використовуються лише два, а третє не дає додаткової інформації і в подальшому не використовується. В методі золотого перетину цільова функція обчислюється в точках інтервалу невизначеності, які розташовані таким чином, щоб кожне обчислене значення цільової функції давало нову корисну інформацію.

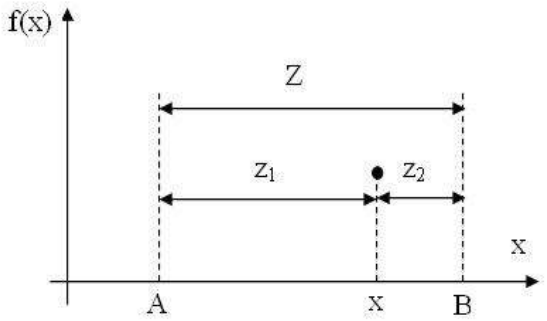


Рис 4.5.3.3 – Позначення, які використовуються в методі золотого перетину

Суть цього методу полягає в наступному. Інтервал невизначеності ділиться на дві нерівні частини, відношення довжини більшого відрізку до довжини всього інтервалу дорівнює відношенню довжини меншого відрізку до довжини більшого відрізку. На рис. 4.5.3.3 показаний інтервал невизначеності Z, який складається з відрізків z_1 і z_2, відношення довжин яких визначається правилом золотого перетину.

$$\frac{z_1}{Z} = \frac{z_2}{z_1}.$$

Крім того, $z_1 + z_2 = Z$. Із першого рівняння витікає $z_1^2 = Z \cdot z_2$. Підставивши значення Z з другого рівняння і поділивши обидві частини на z_1^2, отримаємо

$$\left(\frac{z_2}{z_1}\right)^2 + \frac{z_2}{z_1} - 1 = 0.$$

Розв'язуючи це квадратне рівняння, знаходимо для додатнього кореня значення

$$\frac{z_2}{z_1} = \frac{-1+\sqrt{5}}{2} = 0{,}618.$$

На рис. 5 показано ділення інтервалу невизначеності в цьому відношенні і нанесені відповідні значення цільової функції, які дозволяють зменшити інтервал невизначеності в 1/ 0,618 раз.

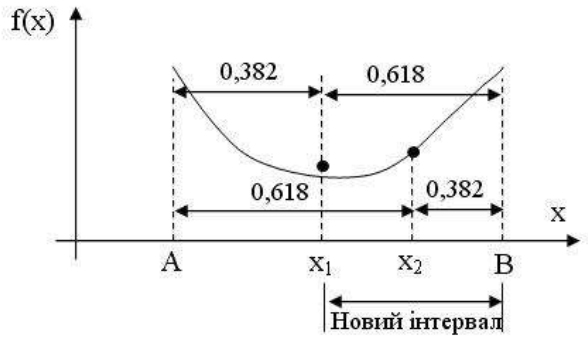


Рис. 4.5.3.4 – Метод золотого перетину

На цій стадії ще не видно переваг методу золотого перетину в порівнянні з методом дихотомії, однак їх добре видно при подальшому діленні інтервалу, так як виявляється, що одне із значень цільової функції, яке необхідно обчислити на наступному кроці, вже відомо. Тому, щоб зменшити невизначеність ще в 1/ 0,618 раз, потрібно додатково обчислити тільки одне значення цільової функції в точці, яка визначається правилом золотого перетину.

При $n > 2$ ефективність методу золотого перетину вища, ніж у метода дихотомії, так як при кожному наступному обчисленні цільової функції інтервал невизначеності скорочується в 1/ 0,618 раз. Після обчислення N значень цільової функції коефіцієнт дроблення інтервалу невизначеності складає

f = 0,618 N-1.

Метод золотого перетину дозволяє відмітити цікаву закономірність: найбільше скорочення наступних інтервалів невизначеності досягається при обчисленні цільової функції в точках, рівновіддалених від його центру.

Якщо продовжувати таким чином і кожного разу, обчислюючи цільову функцію, скорочувати інтервал невизначеності, то будуть справедливими наступні відношення:

$Z_{J-2} = Z_{J-1} + Z_J,\ 1<=""\ n,<=""\ i="">$

де Z_J довжина інтервалу невизначеності після обчислення J-го значення цільової функції.

На рис. 4.5.3.4 представлений алгоритм вибору наступної точки пошуку. Задана точність може, звичайно, змінюватися вибором значення. Для функції $f(x) = -e^{-x}\ln(x)$, пошук відбувався в інтервалі (0, 2).

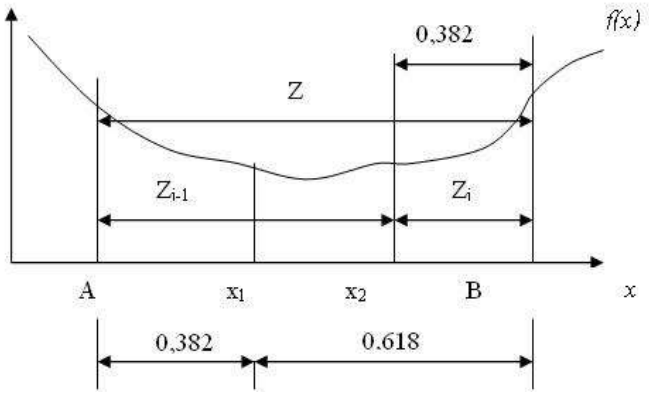


Рис 4.5.3.5 *- Послідовність етапів вибору наступної точки пошуку*

Істинний мінімум знаходиться в точці 1,76322211, де значення функції дорівнює -0, 0972601313.

Метод Кіфера (метод запропонований Кіфером част називають методом Фібоначчі). Припустимо, що потрібно визначити мінімум цільової функції як можна точніше, тобто з найменшим можливим інтервалом невизначеності, але при цьому можна виконати тільки n обчислень функції. Як слід вибрати n точок, в яких обчислюється функція? З першого погляду здається ясним, що не слід шукати рішення

для всіх точок, які одержані в результаті експерименту. Навпаки, треба спробувати зробити так, щоб значення функції, отримані в попередніх експериментах, визначали положення наступних точок. Справді, знаючи значення функції, ми тим самим маємо інформацію про саму функцію і положення її мінімуму і використаємо цю інформацію в подальшому пошуку. Припустимо, що є інтервал невизначеності (x_1, x_3) і відомо значення функції $f(x_2)$ всередині цього інтервалу (див. рис. 4.5.3.7).

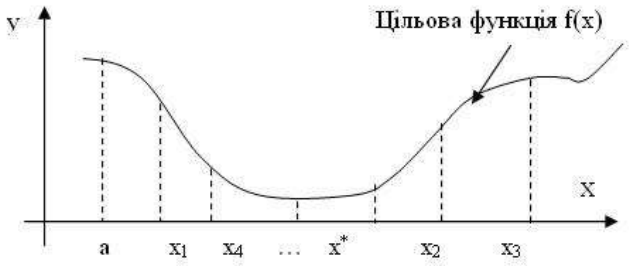


Рис. 4.5.3.7 – Унімодальна цільова функція.

Геометрична інтерпретація метода Фібоначчі

Якщо можна обчислити функцію всього один раз в точці x_4, то де слід помістити точку x_4, для того щоб отримати найменший можливий інтервал невизначеності?

Припустимо $x_2 - x_1 = L$ і $x_3 - x_2 = R$, причому $L > R$, як показано на рис. 13.15, і ці значення будуть фіксовані, якщо відомі x_1, x_2 і x_3. Якщо x_4 знаходиться в інтервалі (x_1, x_2), то:

1. Якщо $f(x_4) < f(x_2)$, то новим інтервалом невизначеності буде (x_1, x_2) довжиною $x_2 - x_1 = L$.

2. Якщо $f(x_4) > f(x_2)$, те новим інтервалом невизначеності буде (x_4, x_3) довжиною $x_3 - x_4$.

Оскільки невідомо, яка з цих ситуацій буде мати місце, виберемо x_4 таким чином, щоб зробити мінімальною найбільшу з довжин $x_3 - x_4$ і $x_2 - x_1$.

Досягнути цього можна, зробивши довжини $x_3 - x_4$ і $x_2 - x_1$ рівними, тобто помістивши x_4 всередині інтервалу симетрично відносно точки x_3, що вже лежить всередині інтервалу. Будь-яке інше положення точки x_4 може призвести до того, що отриманий інтервал буде більший L. Помістивши x_4 симетрично відносно x_2, ми нічим не ризикуємо в будь-якому випадку.

Якщо виявиться, що можна виконати ще одне обчислення функції, то слід застосувати описану процедуру до інтервалу (x_1, x_2), в якому вже є значення функції, обчислене в точці x_3. Отже, стратегія зрозуміла з самого початку. Потрібно помістити наступну точку всередині інтервалу невизначеності симетрично відносно точки, яка вже знаходиться там. Парадоксально, але, щоб зрозуміти, як потрібно починати обчислення, необхідно розібратися в тому, як його потрібно закінчувати.

На n- ому обчисленні n-у точку потрібно помістити симетрично по відношенню до (n-1)-ї точки. Положення цієї останньої точки в принципі залежить від нас. Для того щоб отримати найбільше зменшення інтервалу на даному етапі, слід поділити навпіл попередній інтервал. Тоді точка x_n буде співпадати з точкою x_{n-1}. Однак при цьому ми не одержуємо жодної нової інформації. Звичайно точки x_{n-1} і x_n знаходяться одна від одної на достатній відстані, щоб визначити, в якій половині, лівій чи правій, знаходиться інтервал невизначеності. Вони розміщуються на відстані $\varepsilon/2$ по обидві сторони від середини відрізка L_{n-1}; можна самостійно задати величину ε або вибрати цю величину рівній мінімально можливій відстані між двома точками. (Припустимо, що в нашому прикладі інженер може регулювати температуру з інтервалом в 1С°, тому $\varepsilon = 1$.)

Інтервал невизначеності буде мати довжину L_n, отже, $L_{n-1} = 2L_n - \varepsilon$ (рис. 4.5.3.8, нижня частина).

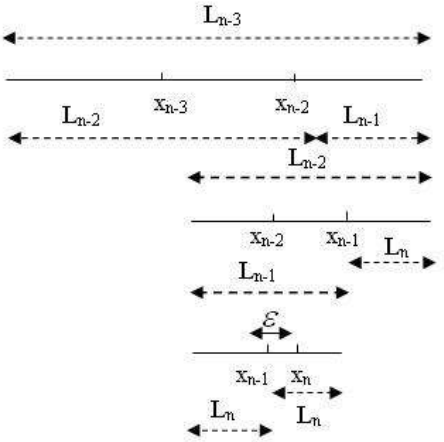


Рис 4.5.3.8 – *Геометрична інтерпретація ітераційного процесу Фібоначчі*

На попередньому етапі точки x_{n-1} і x_{n-2} повинні бути поміщені симетрично всередині інтервалу з L_{n-2} на відстані L_{n-1} від кінців цього інтервалу. Отже, $L_{n-2} = L_{n-1} + L_n$ (рис. 8, середня частина).З рисунку зрозуміло, що на передостанньому етапі x_{n-2} залишається в якості внутрішньої точки.

Аналогічно

$L_{n-3} = L_{n-2} + L_{n-1}$. (рис. 8, верхня частина)

В загальному випадку

$L_{j-1} = L_j + L_{j+1}$ при 1<j<=”” p=””></j

Таким чином,

$L_{n-1} = 2L_n - \varepsilon$,

$L_{n-2} = L_{n-1} + L_n = 3L_n - \varepsilon$,

$L_{n-3} = L_{n-2} + L_{n-1} = 5L_n - 2\varepsilon$,

$L_{n-4} = L_{n-3} + L_{n-2} = 8L_n - 3\varepsilon$ *і т.д.*

Якщо визначити послідовність чисел Фібоначчі наступним чином:

$F_0 = 1,\ F_1 = 1,$ *та* $F_k = F_{k-1} + F_{k-2}$ *для* $k = 2,3,\ldots,$ *тоді*

$L_{n-j} = F_{j+1}L_n - F_{j-1}\varepsilon,\ j = 1,2,\ldots,n-1.$

Якщо початковий інтервал (a,b) має довжину $L_1 = (b-a)$, то $L_1 = F_n L_n - \varepsilon \cdot F_{n-2}$.

Тобто $L_n = \frac{L_1}{F_n} + \varepsilon \frac{F_{n-1}}{F_n}$.

Отже, зробивши n обчислень функції, ми зменшимо початковий інтервал невизначеності в $1/F_n$ раз у порівнянні з його початковою довжиною (нехтуючи ε), і це буде найкращий результат. Якщо ми вже почали пошук, то його нескладно продовжити, використовуючи описане вище правило симетрії. Отже, необхідно знайти положення першої точки, що розміщується на відстані L_2 від одного з кінців початкового інтервалу, причому не важливо, від якого кінця, оскільки друга точка розміщується згідно правила симетрії на відстані L_2 від другого кінця інтервалу:

$$L_2 = F_{n-1} L_n - \varepsilon F_{n-3} = F_{n-1} \frac{L_1}{F_n} + \varepsilon \frac{(F_{n-1} F_{n-2} - F_n F_{n-3})}{F_n} = \frac{F_{n-1}}{F_n} \cdot L_1 + \frac{(-1)^n \varepsilon}{F_n}.$$

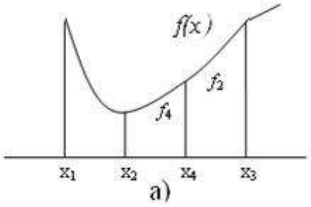
Після того як знайдене положення першої точки, числа Фібоначчі більше не потрібні. Використане значення ε може бути визначене з практичних міркувань. Воно повинне бути менше L_1 / F_{n+1}, в протилежному випадку ми будемо марно витрачати час на обчислення функції.

Таким чином, пошук методом Фібоначчі, названий так через появу при пошуку чисел Фібоначчі, є ітераційною процедурою. В процесі пошуку інтервалу (x_1, x_2) з точкою x_2, що вже лежить в цьому інтервалі, наступна точка x_4 завжди вибирається такою, що $x_3 - x_4 = x_2 - x_1$ або $x_4 - x_1 = x_3 - x_2$, тобто

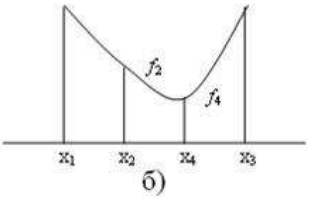
$x_4 = x_1 - x_2 + x_3$.

Якщо $f(x_2) = f_2$ та $f(x_4) = f_4$, тоді можна розглянути чотири випадки (рис. 8).

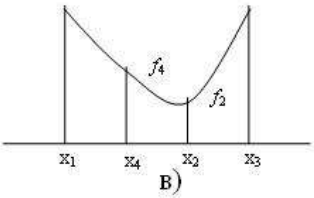
а) $x_4 < x_2$
$f_4 < f_2$
Новий інтервал (x_1, x_2),
що містить точку x_4



б) $x_4 > x_2$
$f_4 < f_2$
Новий інтервал (x_2, x_3),
що містить точку x_4



в) $x_4 < x_2$
$f_4 > f_2$
Новий інтервал (x_4, x_3)
що містить точку x_2



г) $x_4 > x_2$
$f_4 > f_2$
Новий інтервал (x_1, x_4)
що містить точку x_2

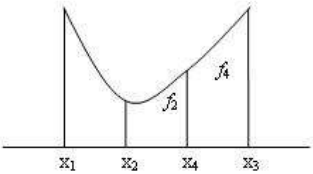


Рис. 4.5.3.9 – Геометрична інтерпретація алгоритму визначення цільової функції на і-му кроці ітерації

Нижче наведено текст програми, що реалізує алгоритм і може бути використаний у якості приклада.

```
double l1=(xmax-xmin),l2,ln,x1=xmin, x2=xmax, x3, x4,
                    trz=(xmax-xmin)/20, to=trz/10, yu = ymin+(ymax - ymin)/20;
               double[] f= new double [30];
               f[1]=1;
```

```
                f[2]=1;
                f[3]=2;
                for (int i=4;i<30;i++){
                        f[i]=f[i-1]+f[i-2];
                }
                int n=3;

                ln=l1*(1/f[n] +(f[n-2]/f[n])*to);
                while (l1*(1/f[n] +(f[n-2]/f[n])*to)>trz){
                    n=n+1;
                }
                l2=(f[n - 1] / f[n]) * l1 + (to * Math.Pow((-1), n)) / f[n];
                x3 = x1 + l2;
                double tmp;
                for (int i=1;i<=n;i++){
                    x4 = x1 + (x2 - x3);
                    if (x4 < x3) { tmp = x3; x3 = x4; x4=tmp;}
                    if (function((float)x4) >= function((float)x3))
                    {
                        x1 = x1;
                        x2 = x4;
                        x3 = x3;
                    }
                    else
                    {
                        x1 = x3;
                        x3 = x4;
                        x2 = x2;
                    }
                    gr.DrawLine(new Pen(Color.DarkGray, 0.01f), (float)x1, (float)yu,
(float)x2, (float)yu);
                    yu = yu + (ymax - ymin) / 20;
                }
                //Ln=L1 (1/Fn +(Fn- 2 / Fn) to).

                this.paint();
```

Найкращими критеріями порівняння методів пошуку, які були описані вище, є їх ефективність і універсальність. Під ефективністю алгоритму розуміють число обчислень функції, необхідне для досягнення необхідного звуження інтервалу невизначеності. Найкращим в цьому відношенні є метод Фібоначчі, а найгіршим – метод загального пошуку. Конструктор не з великим задоволенням використовує метод Фібоначчі, так як при його застосуванні необхідно заздалегідь задавати число обчислень значень функції. Однак він може скористатися методом золотого перетину. Як правило, методи Фібоначчі і золотого перетину, володіють високою ефективністю, найбільш підходять для розв'язку одновимірних унімодальних задач оптимізації.

Універсальність алгоритму означає, що його можна легко застосувати для розв'язку самих різноманітних задач. В цьому відношенні метод Фібоначчі, поступається іншим, так як потребує окремого обчислення положення точок, в яких будуть визначатися значення цільової функції на кожному новому кроці. Цим приходиться розплачуватися за підвищення

ефективності метода. З точки зору універсальності малоефективний метод загального пошуку має по крайній мірі одну перевагу – його можна з успіхом застосовувати і для неунімодальних функцій, якщо вони достатньо гладкі. Нерідко заздалегідь не відомо, чи є розглянута цільова функція унімодальною. В таких випадках слід використати декілька різних алгоритмів і подивитись, чи дають вони усі один і той самий оптимум. Звідси витікає важливий висновок, який слід мати на увазі, розв'язуючи задачі оптимізації: не існує універсального алгоритму, який дозволяв би розв'язувати будь-які задачі. Вирішуючи складні задачі оптимізації, слід користуватися різними методами, так як це дозволяє збільшити долю вигідних розв'язків.

4.5.4. Методи нульового порядку

Задачі визначення екстремумів у багатовимірному випадку істотно ускладняються. Викликають наступні якісно нові сторони розглядуваної задачі:

1. Функція *F (X)* може мати складну форму. Для графічної інтерпретації поверхні прийнято відображати її за допомогою лінії рівня. Лінія рівня – це крива в 2-х мірному перетині простору параметрів, значення функції, на якій константа. поверхня, відповідна залежності *F (X)* може мати: «яри» або «гребни» (рівень поверхні має структуру, що сильно відрізняється від сферичної); «плато; особисті точки типу «сідло». Це не

має себе аналогією в класі одновимірних функцій. («Сідло» – точка гладкої поверхні, біля якої поверхня лежить з різних сторін від своєї касової плоскості. В околицях сідла є 4 інтегральні криві, які входять в особисту точку. Між ними розташовуються інтегральні криві типу гіпербол).

2. Якщо в одновимірному випадку є тільки два можливих напрямки пошуку, то в багатомірному – таких направлених ∞. У зв'язку з цією центральною проблемою пошуку екстремуму багатомірної функції є проблема вибору напрямків пошуку.
3. Змінні $x_1, x_2, \ldots, x_n$ можуть бути взаємозв'язані.
4. У багаторазовому випадку область допустимих значень має безліч форм.

Для порівняння методів використовують графіки ліній рівня

Методи нульового порядку застосовуються у тих випадках, коли з якихось причин визначення градієнта цільової функції неможливе. Вони використовуються у тих випадках, коли функція f_0 задана алгоритмічно, зокрема, коли для обчислення значень функції за тих чи інших значеннях аргументу потрібно провести натурний або числовий експеримент. Ці методи придатні, коли функція f_0 недиференційована або її значення визначаються з похибкою, що призводить до великих неточностей при обчисленні похідних.

Методи прямого пошуку є методами, у яких використовуються лише значення функції. Розглянемо функцію двох змінних. Її лінії постійного рівня наведені на рис. 4.5.4.1, а мінімум знаходиться в точці (u_1^*, u_2^*). Найпростішим методом пошуку є *метод по координатного спуску*. Із точки А ми робимо пошук мінімуму вздовж напрямку осі u_1 і, таким чином, знаходимо точку В, в якій дотична до лінії постійного рівня паралельна осі u_1. Потім, роблячи пошук із точки В у напрямку осі

u_2, отримуємо точку C, роблячи пошук паралельно осі u_1, отримуємо точку D, і т.д. Таким чином, ми приходимо до оптимальної точки. Будь-який із одномірних методів, описаних у попередній главі, може бути використаний тут для пошуку вздовж осі. Таким чином цю ідею можна застосувати для функцій n змінних.

Перші спроби розв'язку оптимізаційних задач без обмежень на основі прямого пошуку пов'язані з використанням одномірних методів оптимізації. Як правило, при реалізації таких методів допустима область визначення показника якості функціонування системи (цільової функції) замінюється дискретною множиною (ґратами) точок простору керованих змінних, а потім використовуються різні стратегії зменшення області, що містить розв'язок задачі. Часто ця процедура виявляється еквівалентною рівномірному пошуку у вузлах ґрат і, отже, непридатною для розв'язку задач із числом змінних, що перевищує 2. Більш корисна ідея полягає у виборі базової точки й оцінюванні значень цільової функції в точках, що оточують базову. Наприклад, при розв'язку задачі із двома змінними можна скористатися квадратним зразком, зображеним на рис. 4.5.4.1.

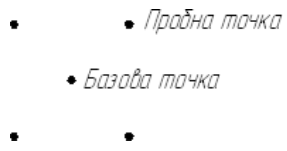


Рис. 4.5.4.1. Квадратний зразок (частковий випадок кубічного зразку)

Потім «найкраща» з п'яти досліджуваних точок вибирається в якості наступної базової точки, навколо якої будується аналогічний зразок. Якщо жодна з кутових точок не має переваги перед базовою, то

розміри зразка варто зменшити, після чого продовжити пошук.

Процедура симплексного пошуку Спендлі, Хекста і Хімсворта базується на тому, що експериментальним зразком, що містить найменшу кількість точок, є регулярний симплекс. Регулярний симплекс в N-мірному просторі є багатогранником, утвореним N+1 рівновіддаленими одна від одної точками-вершинами. Наприклад, у випадку двох змінних симплексом є рівносторонній трикутник; у тривимірному просторі симплекс – тетраедр. В алгоритмі симплексного пошуку використовується важлива властивість симплексів, відповідно до якої новий симплекс можна побудувати на будь-якій грані початкового симплекса шляхом переносу обраної вершини на належну відстань уздовж прямої, проведеної через центр ваги інших вершин початкового симплекса. Отримана в такий спосіб точка є вершиною нового симплекса, а обрана при побудові вершина початкового симплексу відкидається. Неважко побачити, що при переході до нового симплексу потрібно одне обчислення значення цільової функції. Рис. 4.5.4**.2** ілюструє процес побудови нового симплексу на площині.

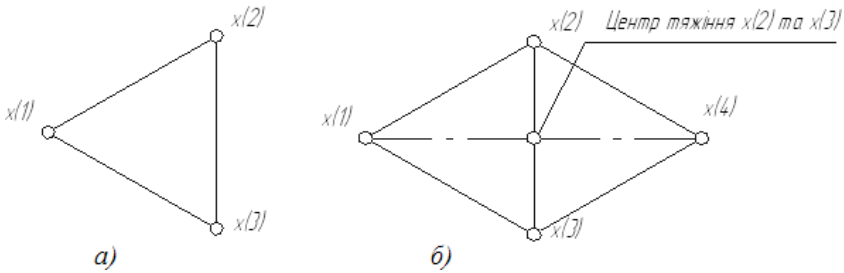


Рис. 4.5.4**.2.** Побудова нового симплексу: а) - початковий симплекс; б) - новий симплекс

Робота алгоритму симплексного пошуку починається з побудови регулярного симплекса у просторі незалежних змінних й оцінювання

значень цільової функції у кожній з вершин симплексу. При цьому визначається вершина, якій відповідає найбільше значення цільової функції. Потім знайдена вершина проектується через центр ваги інших вершин симплексу в нову точку, що використовується як вершина нового симплексу. Якщо функція спадає досить плавно, ітерації тривають доти, поки або не буде накрита точка мінімуму, або не почнеться циклічний рух по двох або більше симплексах. У таких ситуаціях можна скористатися наступними трьома правилами.

Правило 1. «Накриття» точки мінімуму

Якщо вершина, якій відповідає найбільше значення цільової функції, побудована на попередній ітерації, то замість неї береться вершина, якій відповідає наступне по величині значення цільової функції.

Правило 2. Циклічний рух

Якщо деяка вершина симплексу не виключається протягом більш ніж M ітерацій, то необхідно зменшити розміри симплексу за допомогою коефіцієнта редукції і побудувати новий симплекс, обравши в якості базової точку, якій відповідає мінімальне значення цільової функції. Спендлі, Хекст і Хімсворт запропонували обчислювати M по формулі:

$M=1.65N+0.05N^2$,

де N - розмірність задачі, а M округляється до найближчого цілого числа. Для застосування даного правила потрібно встановити величину коефіцієнту редукції.

Правило 3. Критерій закінчення пошуку

Пошук завершується, коли або розміри симплекса, або різниці між значеннями функції у вершинах стають досить малими. Щоб можна було застосовувати ці правила, необхідно задати величину параметру закінчення пошуку.

Реалізація досліджуваного алгоритму заснована на обчисленнях двох типів:

побудові регулярного симплексу при заданих базовій точці й масштабному множнику і розрахунку координат відбитої точки. Побудова симплексу є досить простою процедурою, тому що з елементарної геометрії відомо, що при заданих початковій (базовій) точці x(0) і масштабному множнику координати інших N вершин симплекса в N-мірному просторі обчислюються за простою формулою. Відображення ліній, що демонструють траєкторію спуску симплексу наведено нижче

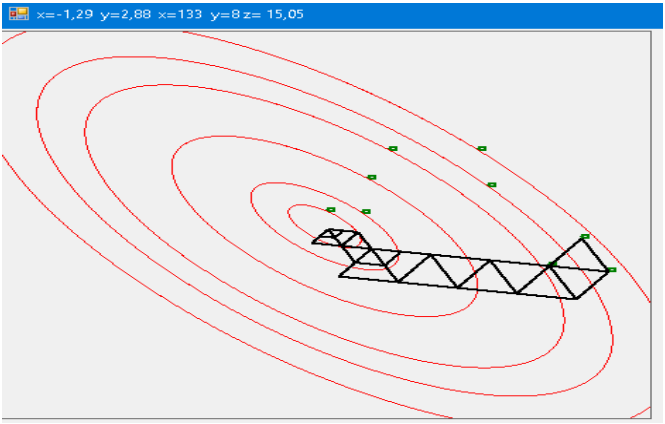


Рис. 4.5.4.3 Траєкторія симплекс методу для еліптичної функції

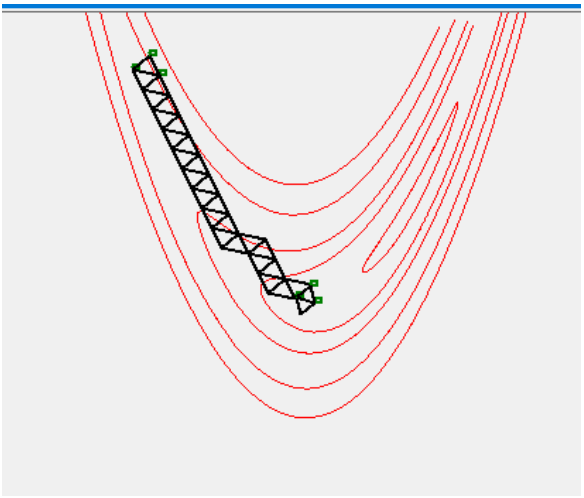


Рис. 4.5.4.4 Траєкторія симплекс методу для функції Розенброка.

З малюнку можна побачити ряд істотних недоліків методу.

1. Можна побачити виникнення труднощів, пов'язаних із масштабуванням, оскільки всі координати вершин симплексу залежать від того самого масштабного множника α. Щоб обійти такі труднощі, у практичних задачах треба про масштабувати всі змінні для того, щоб їх значення були приблизно однаковими по величині.
2. Алгоритм працює занадто повільно, тому що отримана на попередніх ітераціях інформація не використовується для прискорення пошуку.
3. Не існує простого способу розширення симплексу, що не вимагає перерахунку значень цільової функції у всіх точках зразка. Таким чином, якщо з якої-небудь причини цільова функція зменшується (наприклад, якщо зустрічається область із вузьким «яром» або «хребтом»), то пошук повинен тривати із зменшеною величиною кроку.

4.5.5. Методи першого порядку

Всі методи вирішення задач математичного програмування можна розбити на прямі і непрямі.

Непрямі методи вирішення задач математичного програмування є класом алгоритмів, які використовуються для розв'язування оптимізаційних задач без використання явного обчислення градієнту або похідних функції. Ці методи часто застосовуються, коли функція цілі або обмежень є складною або недиференційованою.

Основні характеристики непрямих методів включають:

Використання функцій-оракулів: Непрямі методи отримують інформацію про функцію цілі або обмеження шляхом виклику функцій-оракулів, які повертають значення функції або іншу необхідну інформацію. Це може бути корисно, коли неможливо аналітично отримати похідні або коли обчислення градієнту є обчислювальна складними.

Мінімальна інформація про функцію: Непрямі методи зазвичай працюють з мінімальним набором інформації про функцію, таким як її значення або порівняння значень. Це дозволяє їм ефективно вирішувати задачі, де аналітичні властивості функції не доступні.

Глобальна оптимізація: Багато непрямих методів спрямовані на глобальну оптимізацію, тобто знаходження глобального мінімуму або максимуму функції. Вони намагаються ефективно досліджувати простір параметрів для знаходження найкращого розв'язку.

Можливість роботи з нелінійними обмеженнями: Непрямі методи часто дозволяють вирішувати задачі математичного програмування з нелінійними обмеженнями.

Непрямі методи вирішення задач математичного програмування є класом алгоритмів, які використовуються для розв'язування оптимізаційних задач без використання явного обчислення градієнту або

похідних функції. Ці методи часто застосовуються, коли функція цілі або обмежень є складною або недиференційованою.

Непрямі методи оптимізації. Перелік

Непрямі методи оптимізації включають різні підходи та алгоритми для розв'язання задач математичного програмування без використання градієнту або похідних функції. Ось деякі з найбільш відомих непрямих методів:

Метод дихотомії (метод поділу навпіл): Розділяє область значень на дві частини та повторює процес пошуку в потрібній частині до досягнення заданої точності.

Метод Фібоначчі: Використовує послідовність чисел Фібоначчі для зменшення інтервалу пошуку в кожному кроці.

Метод золотого перетину: Використовує золотий відрізок (відношення золотого перетину) для поділу інтервалу пошуку.

Метод скінченних інтервалів: Розбиває область значень на скінченну кількість інтервалів і обчислює значення функції в кінцях кожного інтервалу.

Метод Нелдера-Міда: Використовує просту геометричну фігуру (симплекс) у просторі параметрів для оцінки та оновлення найкращого розв'язку.

Метод перебору: Використовує перебір всіх можливих комбінацій параметрів для пошуку найкращого розв'язку.

Метод випадкового пошуку: Використовує стохастичний підхід, де нові значення параметрів генеруються випадковим чином.

Метод симуляції відпалу: Використовує аналогію з процесом охолодження металу для пошуку глобального мінімуму.

Генетичні алгоритми: Імітують процес природного відбору та розмноження для еволюції та пошуку оптимального розв'язку.

Методи рою частинок: Метод рою часток, МРЧ (англ. Particle Swarm Optimization, PSO) — метод чисельної оптимізації, для використання якого не потрібно знати точного градієнта оптимізованої функції. МРЧ був доведений Кеннеді, Еберхартом і Ші[1][2] і спочатку призначався для імітації соціальної поведінки. Алгоритм був спрощений, і було зауважено, що він придатний для виконання оптимізації. Книга Кеннеді й Еберхарта [3] описує багато філософських аспектів МРЧ і так званого ройового інтелекту. Велике дослідження застосувань МРЧ зроблене Полі[4][5]. МРЧ оптимізує функцію, підтримуючи популяцію можливих розв'язків, називаних частками, і переміщаючи ці частки в просторі розв'язків згідно із простою формулою. Переміщення підпорядковуються принципу найкращого знайденого в цьому просторі положення, що постійно змінюється при знаходженні частками вигідніших положень.

Метод рою частинок (Particle Swarm Optimization, PSO) - це еволюційний алгоритм, який використовується для глобальної оптимізації. Він був запропонований в 1995 році і здатний вирішувати різноманітні задачі оптимізації.

У методі рою частинок вирішується задача оптимізації шляхом моделювання поведінки "рою" частинок у пошуку оптимального розв'язку. Кожна частинка представляє потенційний розв'язок задачі і рухається у просторі розв'язків залежно від свого власного найкращого розв'язку та найкращого розв'язку, знайденого роєм в цілому.

Процес руху частинок в методі рою частинок включає кілька етапів:

Ініціалізація: Початкові значення частинок (потенційні розв'язки) вибираються випадковим чином у просторі пошуку.

Оцінка функції пристосованості: Кожна частинка оцінюється за допомогою функції пристосованості, яка вимірює якість розв'язку.

Оновлення найкращого розв'язку: Кожна частинка зберігає свій найкращий розв'язок та найкращий розв'язок, знайдений роєм в цілому.

Рух частинок: Кожна частинка рухається у просторі розв'язків на основі свого попереднього руху, власного найкращого розв'язку та найкращого розв'язку, знайденого роєм в цілому. Це рухає частинки в напрямку потенційно більш оптимального розв'язку.

Перевірка критерію зупинки: Алгоритм перевіряє, чи виконано критерій зупинки, який може бути визначений, наприклад, як досягнення мінімуму.

Непрямі методи, засновані на використанні необхідних і достатніх умов екстремуму, і зводяться до вирішення системи нелінійних умов.

Методи, не пов'язані з використанням необхідних і достатніх умов, екстремальних умов відносяться до прямих.

Більшість застосовуваних на практиці прямих методів використовують алгоритми спуска для вирішення задач математичного програмування і вони є ітеративними.

На основі алгоритмів спуску для вирішення задач математичного програмування закладено механізм побудови релаксаційної послідовності елементів простору E^m $\bar{x}^0, \bar{x}^1, \cdots, \bar{x}^k, \bar{x}^{k+1}, \cdots$ за правилами, які визначені у відповідності з методом рішення до якиї належать методи першого порядку. Алгоритми спуска для вирішення задач математичного програмування володіють наступними властивостями:

$$\bar{x}^* = \lim_{i \to \infty} \bar{x}^i$$

$$F(\bar{x}^{k+1}) < F(\bar{x}^k), \quad k = 0,1,\cdots$$

Загальне правило побудови алгоритми спуска для вирішення задач математичного програмування послідовності $\bar{x}^0, \bar{x}^1, \cdots, \bar{x}^k, \bar{x}^{k+1}, \cdots$ записується у вигляді:

$$\bar{x}^{k+1} = \bar{x}^k + \alpha_k \bar{p}^k, \quad k = 0,1,\cdots$$

Тут $\bar{x}^0$ - початкова точка алгоритму, $\bar{p}^k$ - прийнятий на ітерації напрямок переходу з точки $\bar{x}^k$ в точку $\bar{x}^{k+1}$. $\bar{p}^k$ називається напрямком спуска, α^k - числовий множник, що визначає швидкість або величину кроку .

Загальна схема методів градієнтного спуску

Загальна схема методів градієнтного спуску включає наступні кроки:

Ініціалізація: Обирається початкове значення параметрів моделі.

Обчислення градієнта: Обчислюється градієнт функції в поточній точці. Градієнт показує напрямок найшвидшого зростання функції.

Оновлення параметрів: Параметри моделі оновлюються в напрямку, протилежному градієнту. Це здійснюється за допомогою формули: нові_параметри = старі_параметри - крок_оновлення * градієнт.

Перевірка умови зупинки: Перевіряється умова зупинки, яка може бути досягненням заданої точності або досягненням певного числа ітерацій.

Якщо умова зупинки не виконана, повернення до кроку 2. Інакше, оптимізація завершується, і отримані параметри є розв'язком задачі оптимізації.

У методах градієнтного спуску важливими параметрами є крок оновлення (learning rate) і початкове значення параметрів. Вибір правильного кроку оновлення може бути складною задачею, оскільки значення, яке є занадто великим, може призводити до нестійкої збіжності або перепаду, а занадто малий крок може затримувати процес оптимізації. Отже, варто експериментувати з різними значеннями, щоб знайти оптимальний крок оновлення.

Вибір початкової точки $\bar{x}^0$ виробляється, виходячи з фізичного змісту вирішуваної задачі та наявності попередньої інформації про положення точок екстремуму.

Вибір параметрів напрямків і величини кроку визначається стратегією і обраним методом рішення. Критерій перевірки завершення обчислень ітераційного процесу дає інформацію про те, що або рішення завдання має тривати, або знайдена точка, яка претендує на роль екстремуму, і процедуру пошуку слід завершити.

Один з основних критеріїв перевірки завершення обчислень градієнтного спуску - це досягнення певної точки зупинки або заданої точності. Існує кілька широко використовуваних критеріїв, включаючи наступні:

Зупинка за нормою градієнта: Обчислюється норма вектора градієнта і порівнюється з певним заданим порогом. Якщо норма градієнта менша за цей поріг, то вважається, що оптимізація завершена, оскільки знаходиться близько до локального екстремуму.

Зупинка за зміною функціоналу: Обчислюється різниця між значеннями функціоналу (функції, яку мінімізуємо) на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена.

Зупинка за зміною параметрів: Обчислюється різниця між значеннями параметрів на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена.

Зупинка за кількістю ітерацій: Задається максимальна кількість ітерацій, яку необхідно виконати. Якщо ця максимальна кількість досягнута, то вважається, що оптимізація завершена.

Зупинка за зміною величини кроку: Обчислюється різниця між величиною кроку (напрямок спуску) на двох послідовних ітераціях. Якщо ця різниця менша за заданий поріг, то вважається, що оптимізація завершена

Література по розділу 4

1.Kidd, J., Managing with Operations Research (ed.), Heritage Publishers, New Delhi, 1986.

2. Knotts, U.S. (Jr) and E.W. Swift, Management Science for Management 3. Decisions, Allyn and Bacon Inc., Massachussetts, 1978.

3. Kwak, N.K., Mathematical Programming with Business Applications, McGraw-Hill, New York, 1973.

4. Krajewski, L.J. and H.E. Thompson, Management Science, John Wiley & Sons, New York, 1981.

5. Lapin, L., Quantitative Methods for Business Decisions (6th Edn.), The Dryden Press, New York, 1994.

6. Levin, R.I., C.A. Kirkpatrick and D.S. Rubin, Quantitative Approach to Management (5th Edn.), McGraw-Hill, Singapore, 1984.

7. Lee, S.M., L.J. Moore and B.W. Tayler, Management Science, Allyn and Bacon, M.A., 1990.

8. Lee, S.M., Goal Programming for Decision Analysis, Auerbach Publishers, Philadelphia, 1972.

9. Lee, S.M. and L.J. Moore, Introduction to Decision Science, Petroceli/Charter Inc., New York, 1975.

10. Loomba, N.P., Management – A Quantitative Perspective, Macmillan Publishing Company, New York, 1978.

11. Mathur, K., and D. Solow, Management Science, Prentice-Hall, Inc., 1994.

12.Markland, R.E., Topics in Management Science, John Wiley & Sons, 1979.

13. Mitchell, G.H., Operations Research Techniques and Examples, The English Univ. Press Ltd., London, 1972.

14. Nar Singh Deo, System Simulation with Digital Computer, Prentice-Hall of India, New Delhi, 1967.

15. Phillips, D.T., A. Ravindra and J.J. Solberg, Operations Research: Principles and Practice, Wiley, New York, 1976.

16. Raifa, H., Decision Analysis, Reading Mass, Addition-Wesley Publishing Co., Philippines, 1968.

17. Rao, K.V., Management Science, McGraw-Hill Book Co., Singapore, 1986. Saatty, T.L., Mathematical Methods of Operations Research, McGraw-Hill, New York, 1969.

18. Simmons, D.M., Non-linear Programming for Operations Research, Prentice-Hall, Englewood Cliffs, N.J., 1975.

19. Sharma, J.K., Quantitative Techniques for Managerial Decisions, Macmillan India Ltd., New Delhi, 2001.

20. Taha, H.A., Operations Research – An Introduction, Prentice-Hall Inc., New Jersey, 1997.

21. Trueman, R.E., Quantitative Methods for Decision Making in Business, Holt-Saunders, New York, 1981.

22. Thierauf, R.J. and R.L. Klekamp, Decision Making Through Operations Research, John Wiley & Sons, New York, 1970.

23. Turban, E. and J.R. Meredith, Fundamentals of Management Science (3rd Edn.), Business Applications Inc. Taxes, 1985.

24.Vajda, S., Theory of Linear and Non-linear Programming, Longman, London, 1974.

25. Vazsonyi, A. and H.F. Spiror, Quantitative Analysis for Business, Prentice-Hall of India, New Delhi, 1987. Waye, E.L., Quantitative Methods in Accounting, D. Van Nostrand Company, New York, 1980.

26. Wanger, H.M., Principles of Operations Research with Applications to Management Science, Prentice-Hall of India, New Delhi, 1982.

27. Wisniewski, M., Quantitative Methods for Decision Makers, Macmillan India Ltd., New Delhi, 1996.

28.Wiest, J.D. and F.K. Levy, A Management Guide to PERT/CPM, Prentice-Hall Englewood Cliffs, N.J., 1969. Winston, W.I., Operations Research: Applications and Algorithms, Duxbury Press, California, 1994.

29.Zoints, S., Linear and Integer Programming, Prentice-Hall, Englewood Cliffs, N.J., 1974.

30. Zountendijk, G., Mathematical Programming Methods, North-Holland, Amsterdam, 1976.

ДОДАТКИ

```
PROGRAM PRE_MAT_MAX_SUM;
{ Програма перетворення квадратної матриці на основі послідовного
підсумовування модулів всіх членів матриці по рядках і стовпцях і
відповідних перестановок елементів по убуванню цих сум}
uses
  Crt; Var  N,i,j,k,i1,w,j1:integer; v,v1,s1,v3:real;A:array[1..50,1..50] of real;
  B:array[1..50] of real; s:array[1..50] of real; Begin       ClrScr; Write('Задайте
число рівнянь вхідних в систему N='); Read(N); for i:=1 to N do       begin
       WriteLn('Введення коефіцієнтів для',i,'-го рівняння матриці');
for j:=1 to N do             begin Write('Введіть коефіцієнт матриці
A(',i,',',j')=');
Read(A[i,j]);   END; begin Write('Введіть значення для матриці вільних
членів B(',i')='); Read(B[i]);          END;       END;
{ Блок перетворення матриці коефіцієнтів по убуванню суми
абсолютних значень коефіцієнтів по рядках} for i:=1 to N do begin
s1:=0; for j:=1 to N do begin s1:=s1+abs(A[i,j]); s[i]:=s1; end;
end; for i:=1 to N do begin k:=i; for i1:=i+1 to N do begin if s[i1]>s[k] then
k:=i1;
end;  v3:=s[i];s[i]:=s[k];s[k]:=v3; for j:=1 to N do begin
v:=A[i,j];A[i,j]:=A[к,j];A[к,j]:=v; end; v1:=B[i];B[i]:=B[k];B[k]:=v1; end;
{ Блок перетворення матриці коефіцієнтів по убуванню суми
абсолютних значень коефіцієнтів по стовпцях} for j:=1 to N do
begin    s1:=0; for i:=1 to N do begin s1:=s1+abs(A[i,j]); s[j]:=s1;
end; end; for j:=1 to N do begin k:=j; for j1:=j+1 to N do begin if s[j1]>s[k] then
k:=j1; end; v3:=s[j];s[j]:=s[k];s[k]:=v3; for i:=1 to N do begin
v:=A[i,j];A[i,j]:=A[i,k];A[i,k]:=v; end; end; for i:=1 to N do begin
for j:=1 to N do   begin  Write(A[i,j]:4:0);  end;  WriteLn(B[i]:4:0); end;
END.
```

```
PROGRAM PRE_MATR_VGE;
{ Програма перетворення квадратної матриці на основі принципу вибору
головного(найбільшого по модулю) елементу з подальшою перестановкою
рядків і стовпців}uses Crt; Var N,i,j,i5,w,w1:integer; v,v1,max:real;
A:array[1..50,1..50] of real; B:array[1..50] of real;Begin ClrScr;
Write('Задайте число рівнянь вхідних в систему N='); Read(N); for i:=1 to N
do begin WriteLn('Введення коефіцієнтів для',i,'-го рівняння матриці'); for
j:=1 to N do begin Write('Введіть коефіцієнт матриці A(',i,',',j')=');
Read(A[i,j]); END; begin Write('Введіть значення для матриці вільних
членів B(',i')='); Read(B[i]); END; END; for i5:=1 to N-1 do begin
max:=abs(A[i5,i5]);w:=i5;w1:=i5; for i:=i5 to N do begin for j:=i5 to N do
begin if abs(A[i,j]) >max then begin w:=i;w1:=j;max:=abs(A[w,w1]);
end; end; end;
{ Блок перестановки рядків матриці по убуванню залежно від значення
модуля головного елементу на кожному кроці} for j:=1 to N do begin
v:=A[i5,j];A[i5,j]:=A[w,j];A[w,j]:=v; end;
v1:=B[i5];B[i5]:=B[w];B[w]:=v1;
{ Блок перестановки стовпців матриці по убуванню залежно від значення
модуля головного елементу на кожному кроці} for i:=1 to N do begin
v:=A[i,i5];A[i,i5]:=A[i,w1];A[i,w1]:=v; end; end; for i:=1 to N do begin
for j:=1 to N do begin Write(A[i,j]:4:0); end; WriteLn(B[i]:4:0); end;
END.
```

```
PROGRAM S_G_PE;
USES Crt;
{ Вирішення системи з N лінійних рівнянь методом Гауса послідовного
виключення невідомих}
var N,i,j,u,m,k,z:integer; v,v1,H:real; A:array[1..50,1..50] of real;
  B:array[1..50] of real;  X:array[1..50] of real; begin ClrScr;
```

```
 Write('Задайте число рівнянь вхідних в систему N='); Read(N); for i:=1 to N
do    begin  WriteLn('Введення коефіцієнтів для',i,'-го рівняння матриці');for
j:=1 to N do  begin Write('Введіть коефіцієнт матриці A(',i,',',j')=');
 Read(A[i,j]); END; begin
 Write('Введіть значення для матриці вільних членів B(',i')='); Read(B[i]);
 END;END;{ begin u:=1; for m:=1 to N do begin for i:=1*u to N do  begin
 k:=i; for j:=i+1 to N do   if abs(A[j,m]) >abs(A[к,m]) then k:=j;
 for z:=1 to N do begin v:=A[i,z];A[i,z]:=A[к,z];A[к,z]:=v;END;
v1:=B[i];B[i]:=B[k];B[k]:=v1;   END;   u:=u+1;  END; END;}
 for i:=1 to N-1 do  begin   for j:=i+1 to N do  begin   A[j,i]:=-A[j,i]/A[i,i];
 for k:=i+1 to N do   begin   A[j,k]:=A[j,k]+A[j,i]*A[i,k];  END;
 B[j]:=B[j]+A[j,i]*B[i];  END;  END;   X[N]:=B[N]/A[N,N];  for i:=N-1 downto
1 do
 begin   H:=B[i];   for j:=i+1 to N do   begin   H:=H-X[j]*A[i,j];   END;
 X[i]:=H/A[i,i];   END;  WriteLn('Коріння системи лінійних рівнянь');
 for i:=1 to N do  begin  WriteLn('X(',i')=',X[i]:10:4);  END;  END.
```

```
                    PROGRAM S_G_VGE;
USES   Crt;
{ Вирішення системи з N лінійних рівнянь методом Гауса з вибором
головного елементу}
var   N,i,j,u,m,k,z,N1,k1,L,j1:integer;   v,v1,H,S:real;   A:array[1..50,1..50] of
real;
B:array[1..50] of real;  X:array[1..50] of real; C:array[1..50,1..50] of real;
G:array[1..50] of real; begin  ClrScr;
Write('Задайте число рівнянь вхідних в систему N=');  Read(N);
for i:=1 to N do begin WriteLn('Введення коефіцієнтів для',i,'-го рівняння
матриці');  for j:=1 to N do  begin  Write('Введіть коефіцієнт матриці
A(',i,',',j')=');  Read(A[i,j]);   END;   begin
 Write('Введіть значення для матриці вільних членів B(',i')=');
```

```
Read(B[i]); END; END; begin u:=1; for m:=1 to N do begin
for i:=1*u to N do begin k:=i; for j:=i+1 to N do if abs(A[j,m]) >abs(A[к,m])
then k:=j; for z:=1 to N do begin v:=A[i,z];A[i,z]:=A[к,z];A[к,z]:=v; END;
v1:=B[i];B[i]:=B[k];B[k]:=v1; END; u:=u+1; END; END; N1:=N-1; for k:=1
to N1 do begin if abs(A[до, до]) >0 then begin G[k]:=B[k]/A[до,
до];k1:=k+1;
for i:=k1 to N do begin B[i]:=B[i]-A[i,k]*G[k]; for j1:=k to N do
begin j:=N-j1+k;C[к,j]:=A[к,j]/A[до, до]; A[i,j]:=A[i,j]-A[i,k]*C[к,j];
END; END; END else begin k1:=k+1; for m:=k1 to N do
begin if abs(A[m,k]) >0 then for L:=1 to N do begin
v:=A[к,L];A[к,L]:=A[m,L];A[m,L]:=v; END; END;
begin v:=B[k];B[k]:=B[m];B[m]:=v; G[k]:=B[k]/A[до, до];k1:=k+1;
for i:=k1 to N do begin B[i]:=B[i]-A[i,k]*G[k];
for j1:=k to N do begin j:=N-j1+k;C[к,j]:=A[к,j]/A[до, до];
A[i,j]:=A[i,j]-A[i,k]*C[к,j]; END; END;END; END; END;
m:=N;X[m]:=B[m]/A[m,m]; for m:=N-1 downto 1 do begin
S:=0; for L:=m to N1 do begin S:=S+C[m,L+1]*X[L+1]; END;
X[m]:=G[m]-S; END; for i:=1 to N do begin WriteLn('X(',i')=',X[i]:10:4);
END;END.
```

```
PROGRAM S_I_PRO;
USES Crt;
{ Вирішення системи з N лінійних рівнянь методом простих ітерацій}
var N,i,j,u,k,z1,m:integer; S,E,v,v1:real; A:array[1..50,1..50] of real;
B:array[1..50] of real; X:array[1..50] of real; z:array[1..50] of real;
BEGIN ClrScr; Write('Задайте число рівнянь вхідних в систему N=');
Read(N); Write('Задайте погрішність обчислення E='); Read(E);
for i:=1 to N do begin WriteLn('Введення коефіцієнтів для',i,'-го рівняння
матриці'); for j:=1 to N do begin Write('Введіть коефіцієнт матриці
A(',i,',',j')='); Read(A[i,j]); END; begin Write('Введіть значення для
```

```
матриці вільних членів B(',i')='); Read(B[i]); END; END; begin u:=1; for
m:=1 to N do
begin for i:=1*u to N do begin k:=i; for j:=i+1 to N do begin if abs(A[j,m])
>abs(A[к,m]) then k:=j; END; for z1:=1 to N do begin
v:=A[i,z1];A[i,z1]:=A[к,z1];A[к,z1]:=v; END; v1:=B[i];B[i]:=B[k];B[k]:=v1;
END; u:=u+1; END; END; S:=0; for i:=1 to N do begin
WriteLn('Задайте початкове наближення для кореня XO(',i')=');
Read(z[i]); end; repeat k:=0; for i:=1 to N do begin X[i]:=-B[i];
for j:=1 to N do begin X[i]:=X[i]+A[i,j]*z[j]; end;
if abs(X[i]/A[i,i]) >=E then k:=1; X[i]:=z[i]-(X[i]/A[i,i]); end; for i:=1 to N do
begin z[i]:=X[i]; end; S:=S+1; until k<>1; WriteLn('Коріння системи
лінійних рівнянь'); for i:=1 to N do begin WriteLn('X(',i')=',X[i]:10:4); end;
WriteLn('Число итераций',S:10:0); END.
```

```
PROGRAM S_I_ZEID;
USES
  Crt;{ Вирішення системи з N лінійних рівнянь методом Зейделя}
var N,i,j,u,k,z1,m:integer; S,E,v,v1:real; A:array[1..50,1..50] of real;
B:array[1..50] of real; X:array[1..50] of real; z:array[1..50] of real;
BEGIN ClrScr; Write('Задайте число рівнянь вхідних в систему N=');
Read(N); Write('Задайте погрішність обчислення E='); Read(E);
for i:=1 to N do begin WriteLn('Введення коефіцієнтів для',i,'-го рівняння
матриці'); for j:=1 to N do begin Write('Введіть коефіцієнт матриці
A(',i,',',j')='); Read(A[i,j]); END; begin
Write('Введіть значення для матриці вільних членів B(',i')=');
 Read(B[i]); END; END; begin u:=1; for m:=1 to N do
 begin for i:=1*u to N do begin k:=i; for j:=i+1 to N do
 begin if abs(A[j,m]) >abs(A[к,m]) then k:=j; END; for z1:=1 to N do
```

```
 begin  v:=A[i,z1];A[i,z1]:=A[к,z1];A[к,z1]:=v;   END;
v1:=B[i];B[i]:=B[k];B[k]:=v1;  END;  u:=u+1;  END;  END;   S:=0;for i:=1 to N
do  begin  WriteLn('Задайте початкове наближення для кореня X0(',i')=');
Read(z[i]);  end; repeat  k:=0; for i:=1 to N do  begin  X[i]:=-B[i];
for j:=1 to N do  begin  X[i]:=X[i]+A[i,j]*z[j];  end; if abs(X[i]/A[i,i]) >=E then
k:=1;
X[i]:=z[i]-(X[i]/A[i,i]);  z[i]:=X[i];  end;  S:=S+1;  until k<>1;
WriteLn('Коріння системи лінійних рівнянь'); for i:=1 to N do
begin  WriteLn('X(',i')=',X[i]:10:4);  end; WriteLn('Число итераций',S:10:0);
END.
```

```
PROGRAM S_M_VRAS;
USES  Crt;
{ Вирішення системи з N лінійних рівнянь методом обертання}
var  N,i,j,u,m1,k,z:integer; L,R,v,v1,m:real; A:array[1..50,1..50] of real;
B:array[1..50] of real; B1:array[1..50] of real;BEGIN  ClrScr;
Write('Задайте число рівнянь вхідних в систему N=');  Read(N);m:=0;
for i:=1 to N do  begin
WriteLn('Введення коефіцієнтів для',i,'-го рівняння матриці');
for j:=1 to N do  begin Write('Введіть коефіцієнт матриці A(',i,',',j')=');
Read(A[i,j]);  END;  begin
Write('Введіть значення для матриці вільних членів B(',i')=');
Read(B[i]);  END; END;  begin  u:=1;  for m1:=1 to N do begin for i:=1*u to N
do
begin  k:=i; for j:=i+1 to N do  begin if abs(A[j,m1]) >abs(A[к,m1]) then k:=j;
END;  for z:=1 to N do  begin  v:=A[i,z];A[i,z]:=A[к,z];A[к,z]:=v;  END;
v1:=B[i];B[i]:=B[k];B[k]:=v1;  END;  u:=u+1;  END; END;  for i:=1 to N-1 do
begin  for k:=i+1 to N do  begin   if A[i,i]=A[к,i] then  begin  if A[к,i]=0 then
begin   m:=1;L:=0;  end  else  begin  m:=sqrt((A[i,i]*A[i,i])+(A[к,i]*A[к,i]));
```

```
L:=-A[к,i]/m;m:=A[i,i]/m; end;  end  else  begin
m:=sqrt((A[i,i]*A[i,i])+(A[к,i]*A[к,i]));  L:=-A[к,i]/m;m:=A[i,i]/m;
end;  for j:=1 to N do  begin  R:=(m*A[i,j])-L*A[к,j];
A[к,j]:=L*A[i,j]+(m*A[к,j]);
A[i,j]:=R;  end;  R:=(m*B[i])-L*B[k];  B[k]:=(L*B[i])+m*B[k];
B[i]:=R;  end;  end;  for i:=N downto 1 do  begin  m:=0;  for k:=0 to N-i-1 do
begin  m:=m+B1[N-k]*A[i,N-k];  end; B1[i]:=(B[i]-m)/A[i,i];
WriteLn('X(',i')=',B1[i]:10:3); end;END.
```

```
PROGRAM KOR_POLIN;
{ Визначення всього коріння полінома An(X^n)+An-1(X^N-
1)+...+A1X+Ao=0
з дійсними коефіцієнтами}USES   Crt;    var
e,C,T,A1,S,R,P,Q,H,M,L,U,D,V,Y,Z,X,B,W,F:real;  N,j,K:integer;
A:array[1..50] of Real;procedure V1; begin  N:=N-2;A1:=0;
if sqrt((S-(R*C/2))*(S-(R*C/2))+R*R*abs(H)) <=e then   begin  A1:=1;   end;
B:=0;  if N>=2 then  begin  B:=1;  end;end;procedure V2;begin P:=-C/2;
Q:=sqrt(abs(H));  if H>=0 then  begin    M:=P+Q;P:=P-Q;Q:=0;   end  else
begin   M:=P;  end;end;procedure V3;label M1;begin  begin  repeat  begin
repeat M1:  if M<>10 then   begin  M:=M+1; H:=0;Q:=A[1];  P:=A[2]-C*Q;
L:=Q;
end  else  begin   P:=C; M:=0; Q:=D;  C:=U; D:=V;  U:=P;V:=Q;Y:=C;  Z:=D;
F:=-F;
M:=M+1; H:=0; Q:=A[1];   P:=A[2]-C*Q; L:=Q;  end;  for j:=3 to N do  begin
R:=P; P:=A[j]-C*R-D*Q;  Q:=R; R:=L; L:=Q-C*R-H*D;  H:=R;  end;
Q:=A[N+1]-D*Q; S:=L+C*R;   if T=0 then   begin  X:=D*R; H:=R*X+S*L;
if H=0 then  begin {Continue;}  goto M1;  end;  end;  until True;  end;
C:=C+((P*S-Q*R)/H); D:=D+((P*X+Q*L)/H);  if (C-Y+D-Z) <>0 then
begin  end  else  begin  if F=-W then  begin  WriteLn('Немає рішення');
Exit;  end  else begin  W:=-F;  end;  end;  H:=(C*C/4) -D;
```

```
if sqrt((Q-(P*C/2))*(Q-(P*C/2))+P*P*abs(H)) >e/N then  begin { Continue;}
goto M1;  end; until True; end;  T:=0; A[2]:=A[2]-C*A[1]; for j:=3 to N-1 do
begin  A[j]:=A[j]-C*A[j-1]-D*A[j-2];  end;  V2;end;procedure V4;begin
for j:=3 to N do  begin  R:=P; P:=A[j]-C*R-D*Q;  Q:=R; R:=L; L:=Q-C*R-
H*D;
H:=R;  end; Q:=A[N+1]-D*Q; S:=L+C*R; if T=0 then  begin  X:=D*R;
H:=R*X+S*L;  if H=0 then  begin  V3;  end;  end;
C:=C+((P*S-Q*R)/H); D:=D+((P*X+Q*L)/H);  if (C-Y+D-Z) <>0 then
begin end  else  begin  if F=-W then  begin  WriteLn('Немає рішення');
Exit;  end  else begin  W:=-F;  end;  end;  H:=(C*C/4) -D;
if sqrt((Q-(P*C/2))*(Q-(P*C/2))+P*P*abs(H)) >e/N then  begin  V3;  end;
T:=0; A[2]:=A[2]-C*A[1];  for j:=3 to N-1 do  begin  A[j]:=A[j]-C*A[j-1]-
D*A[j-2];
end;  V2;end;begin  ClrScr; WriteLn('Задайте погрішність визначення
коріння');  Write('e=');  ReadLn(e);  WriteLn('Задайте ступінь полінома');
Write('N=');  ReadLn(N);  WriteLn('Введіть коефіцієнти полінома');
for j:=1 to N+1 do  begin  Write('A(',N+1-j')='); ReadLn(A[j]);  end; K:=1;
repeat  T:=1;C:=A[2]/A[1];  if N=1 then  begin  P:=-C; Q:=0;
WriteLn('X(',K')=',P:12:5,'+j*(',-Q:12:5',)');  K:=K+1;  if T<>0 then Break;
V1;  end else begin if N=2 then  begin  H:=(C*C/4) -(A[3]/A[1]);
V2;  WriteLn('X(',K')=',M:12:5,'+j*(',Q:12:5')');  K:=K+1;
WriteLn('X(',K')=',P:12:5,'+j*(',-Q:12:5',)');  K:=K+1;  if T<>0 then Break;
V1;  end  else  begin  M:=10; C:=4; D:=8; U:=4;  V:=8; F:=1; W:=2; T:=0;
V3;  WriteLn('X(',K')=',M:12:5,'+j*(',Q:12:5')');  K:=K+1;
WriteLn('X(',K')=',P:12:5,'+j*(',-Q:12:5',)');  K:=K+1;  if T<>0 then Break;
V1;  end;  end;  if A1+B=2 then  begin  T:=1; V4;
WriteLn('X(',K')=',M:12:5,'+j*(',Q:12:5')');  K:=K+1;
WriteLn('X(',K')=',P:12:5,'+j*(',-Q:12:5',)');  K:=K+1;  if T<>0 then Break;V1;
end;  until False;END.
```

```
PROGRAM SU_TR_U_M_N_M;
{ Вирішення системи трансцендентних рівнянь Fi(Xi)=0 модифікованим
методом Ньютона}USES Crt; const
l1=300;l3=400;l2=400;h1=100;co=4;ct=28;n1=1;n2=7;lca=316.228; var
e,x1,h,lao,la1o,lod,lcao,lca1o,lodt,p1,p2,pt:Real; N,M,s,i,j,k,R:integer;
A:array[1..50,1..50] of Real; B:array[1..50] of Real; x:array[1..50] of Real;
F:array[1..50] of Real; procedure V;
{ Представляємо трансцендентні рівняння у вигляді fi(Xi)=0 і
підставляємо праві частини отриманих рівнянь у вирази
F(i)=f(Xi)} begin { Значення розтягнутої ділянки основи до ремізної рамки
lca:=sqrt(sqr(l1)+sqr(h1));} lao:=sqrt(sqr(l2-x[1])+sqr(h1-x[2]));
la1o:=sqrt(sqr(l2-x[1])+sqr(h1+x[2])); lod:=sqrt(sqr(l3+x[1])+sqr(x[2]));
lcao:=lca+lao-(l1+l2);lca1o:=lca+la1o-(l1+l2);lodt:=lod-l3;
p1:=n2*co*lcao;p2:=n1*co*lca1o;pt:=ct*lodt;
F[1]:=(-p1*(l2-x[1])/lao)+(-p2*(l2-x[1])/la1o)+(pt*(l3+x[1]) /lod);
F[2]:=(p1*(h1-x[2]) /lao)+(-p2*(h1+x[2])/la1o)+(-pt*x[2]/lod); end;begin
ClrScr; WriteLn('Задайте число рівнянь'); Write('N='); ReadLn(N);
WriteLn('Задайте максимальне число ітерацій'); Write('M='); ReadLn(M);
WriteLn('Задайте відносну погрішність'); Write('e='); ReadLn(e);
s:=0; WriteLn('Введіть початкові значення'); for i:=1 to N do begin
Write('X',i'(0)='); ReadLn(x[i]); end; repeat V; for i:=1 to N do begin
B[i]:=-F[i]; end; for j:=1 to N do begin x1:=x[j]; h:=e*abs(x1); x[j]:=x1+h;
V; for i:=1 to N do begin A[i,j]:=(F[i]+B[i])/h; end; x[j]:=x1; end; s:=s+1;
if s=M+1 then begin Write('Число ітерацій дорівнює s=',s); Break;
end; for i:=1 to N-1 do begin for j:=i+1 to N do begin A[j,i]:=-A[j,i]/A[i,i];
for k:=i+1 to N do begin A[j,k]:=A[j,k]+A[j,i]*A[i,k]; end;
B[j]:=B[j]+A[j,i]*B[i]; end; end; F[N]:=B[N]/A[N,N]; for i:=N-1 downto 1
do
begin h:=B[i]; for j:=i+1 to N do begin h:=h-F[j]*A[i,j]; end; F[i]:=h/A[i,i];
```

```
end;  R:=0;  for i:=1 to N do  begin   x[i]:=x[i]+F[i];  if abs(F[i]/x[i]) >e then
R:=1;
end;  if R=1 then Continue;  WriteLn('Коріння вирішення системи рівнянь');
for i:=1 to N do  begin  WriteLn('X(',i')=',x[i]:14:9);  end;
WriteLn('Число ітерацій s=',s);  Break; until false;END.
```

```
PROGRAM SU_TR_U_M_N_M;
{ Вирішення системи трансцендентних рівнянь  Fi(Xi)=0 модифікованим
методом Ньютона}USES   Crt;   var   e,x1,h:Real;   N,M,s,i,j,k,R:integer;
A:array[1..50,1..50] of Real;   B:array[1..50] of Real;   x:array[1..50] of Real;
F:array[1..50] of Real;   procedure V;{ Представляємо трансцендентні
рівняння у вигляді fi(Xi)=0 і підставляємо праві частини отриманих
рівнянь у вирази
F(i)=f(Xi)}   begin   F[1]:=x[1]+3*(Ln(x[1]) /Ln(10))-x[2]*x[2];
F[2]:=2*x[1]*x[1]-x[1]*x[2]-5*x[1]+1;   end;begin   ClrScr;
WriteLn('Задайте число рівнянь');   Write('N=');   ReadLn(N);
WriteLn('Задайте максимальне число ітерацій');   Write('M=');   ReadLn(M);
WriteLn('Задайте відносну погрішність');   Write('e=');   ReadLn(e);   s:=0;
WriteLn('Введіть початкові значення');   for i:=1 to N do      begin
Write('X',i'(0)=');      ReadLn(x[i]);      end;  repeat  V; for i:=1 to N do  begin
B[i]:=-F[i];   end;   for j:=1 to N do   begin
x1:=x[j]; h:=e*abs(x1);   x[j]:=x1+h;  V;  for i:=1 to N do   begin
A[i,j]:=(F[i]+B[i])/h;  end;  x[j]:=x1;  end;  s:=s+1;  if s=M+1 then  begin
Write('Число ітерацій дорівнює s=',s);  Break;  end;  for i:=1 to N-1 do
begin  for j:=i+1 to N do  begin   A[j,i]:=-A[j,i]/A[i,i]; for k:=i+1 to N do
 begin  A[j,k]:=A[j,k]+A[j,i]*A[i,k]; end;  B[j]:=B[j]+A[j,i]*B[i];  end; end;
F[N]:=B[N]/A[N,N];  for i:=N-1 downto 1 do  begin  h:=B[i];  for j:=i+1 to N
do
begin  h:=h-F[j]*A[i,j];   end;  F[i]:=h/A[i,i];  end;  R:=0;  for i:=1 to N do
begin
```

```
x[i]:=x[i]+F[i]; if abs(F[i]/x[i]) >e then R:=1; end; if R=1 then Continue;
WriteLn('Коріння вирішення системи рівнянь'); for i:=1 to N do begin
WriteLn('X(',i')=',x[i]:14:9); end; WriteLn('Число ітерацій s=',s); Break;
until false;END.
```

```
PROGRAM TR_U_M_DI;
{ Вирішення трансцендентного рівняння F(X)=0 методом
дихотомії(ділення
відрізання навпіл)}USES Crt; var x,x1,a,b,e,F:real;
n:integer;procedure V;
{ Представляємо трансцендентне рівняння у вигляді F(x)=0 і
підставляємо ліву частину отриманого рівняння у вираз F=f(x)} begin
x1:=a; x:=(a+b) /2;{Ліва частина підставляється із знаком "+" при
аргументі x1=a} F:=x1-cos(x1); if F>=0 then begin{Ліва частина
підставляється із знаком "+" } F:=x-cos(x); end else begin {Ліва частина
підставляється із знаком "-" } F:=-(x-cos(x)); end; end;begin ClrScr;
WriteLn('Задайте початкову координату інтервалу'); Write('a=');
ReadLn(a); WriteLn('Задайте кінцеву координату інтервалу'); Write('b=');
ReadLn(b); WriteLn('Задайте погрішність обчислення результату');
Write('e='); ReadLn(e); n:=0; repeat
V; if F>0 then begin a:=x; end else begin b:=x; end; n:=n+1; until b-a<e;
WriteLn('Корінь рівняння рівний'); WriteLn('X=',x:12:9);
WriteLn('Число кроків для визначення кореня рівняння із заданою
точністю');
WriteLn('n=',n);END.
```

```
PROGRAM TR_U_M_NJ;
{ Вирішення трансцендентного рівняння X=F(X) методом Ньютона}USES
Crt; var x0,x,e,F,F1,V3:real; n:integer; procedure V;
{ Представляємо трансцендентне рівняння у вигляді f(x)=0 і
підставляємо ліву частину отриманого рівняння у вираз
```

```
F=f(x)}  begin  F:=x-cos(x);  end;  procedure V1;
{ Визначаємо першу похідну рівняння f(x) і підставляємо
у вираз F1=df(x) /dx}  begin  F1:=1+sin(x);  end;begin  ClrScr;
WriteLn('Задайте початкове значення кореня рівняння');  Write('Xo=');
ReadLn(x); WriteLn('Задайте погрішність обчислення результату');
Write('e=');
ReadLn(e); n:=0;  V; V1;  V3:=F/F1;  while  abs(V3) >=e do begin x:=x-V3;
V; V1;  V3:=F/F1;  n:=n+1;  end; WriteLn('Корінь рівняння рівний');
WriteLn('X=',x:12:9); WriteLn('Число кроків для визначення кореня
рівняння із заданою точністю');  WriteLn('n=',n);END.
```

```
PROGRAM TR_U_M_N_M;
{ Вирішення трансцендентного рівняння F(X)=0 модифікованим методом
Ньютона}USES  Crt;  var  x0,x,e,F,L:real;  n:integer;  procedure V;
{ Представляємо трансцендентне рівняння у вигляді x=f(x) і
підставляємо праву частину отриманого рівняння у вираз  F=f(x)} begin
F:=x-cos(x);  end;  procedure V1; begin L:=F; x:=x+e; end;  procedure V2;
begin  L:=e*L/(F-L);  x:=x-L-e;  end;begin  ClrScr;
WriteLn('Задайте початкове значення кореня рівняння'); Write('Xo=');
ReadLn(x);
WriteLn('Задайте погрішність обчислення результату'); Write('e=');
ReadLn(e);
n:=0; V;  V1;  V;  V2;  while  abs(L) >=e do  begin  V;  V1;  V;  V2;  n:=n+1;
end;
WriteLn('Корінь рівняння рівний');  WriteLn('X=',x:12:9);
WriteLn('Число кроків для визначення кореня рівняння із заданою
точністю');
WriteLn('n=',n);END.
```

```
PROGRAM TR_U_M_PI;
```

```
{ Вирішення трансцендентного рівняння X=F(X) методом простих
ітерацій}
USES Crt; var x0,x,e,F:real; n:integer; procedure V;
{ Представляємо трансцендентне рівняння у вигляді x=f(x) і
підставляємо праву частину отриманого рівняння у вираз
F=f(x)} begin F:=sin(x) +0.25; end; procedure V1; begin x:=F; end;
begin ClrScr; WriteLn('Задайте початкове значення кореня рівняння');
Write('Xo='); ReadLn(x); WriteLn('Задайте погрішність обчислення
результату'); Write('e='); ReadLn(e); n:=0; V; while abs(F-x) >=e
do
begin V1; V; n:=n+1; end; WriteLn('Корінь рівняння рівний');
WriteLn('X=',x:12:9);
WriteLn('Число кроків для визначення кореня рівняння із заданою
точністю');
WriteLn('n=',n);END.
```

```
PROGRAM TR_U_M_PP;
{ Визначення всього коріння трансцендентного рівняння F(x)=0 методом
порозрядного наближення на інтервалі a,b}USES Crt; var
x,a,b,e,F,h,c:real; k:integer; w:LongInt; procedure V;
{ Представляємо трансцендентне рівняння у вигляді F(x)=0 і
підставляємо ліву частину отриманого рівняння у вираз
F=f(x)} begin F:=x-cos(x); end;begin ClrScr;
WriteLn('Задайте початкову координату інтервалу'); Write('a=');
ReadLn(a);
WriteLn('Задайте кінцеву координату інтервалу'); Write('b='); ReadLn(b);
WriteLn('Задайте погрішність обчислення результату'); Write('e=');
ReadLn(e);
```

```
WriteLn('Задайте крок початкового пошуку'); Write('h='); ReadLn(h); c:=h;
k:=0; x:=a; V; w:=Trunc(F/abs(F)); repeat x:=x+c; if x-c>=b then Break;
V;
if F*w/c>0 then Continue; c:=-c/4; if abs(c)>(e/4) then Continue; k:=k+1;
WriteLn('X(',k')=',x:12:8); c:=h; w:=-w; until False;END.
                    PROGRAM DIF_UR_SIS_M_RKM_AV;
{ Вирішення системи лінійних диференціальних рівнянь
методом Рунге-кутта-мерсона з автоматичним вибором кроку} USES
Crt; var N,j,d1: integer; x,h,xm,e1,e2,e3: real; y:array[1..50] of real;
w:array[1..50] of real; c:array[1..50] of real; F:array[1..50] of real;
a:array[1..50] of real; d:array[1..50] of real; e:array[1..50] of real;
Procedure V1;
{ Представляємо систему диференціальних рівнянь dyi/dx=fi(x,yi)
 ввиде F(i)=fi(x,y)} Begin F[1]:=y[1]+y[2]-x*x+x-2; F[2]:=-
2*y[1]+4*y[2]+2*x*x-4*x-7; end;begin ClrScr;
WriteLn('Задайте число диференціальних рівнянь першого порядку');
Write('N='); ReadLn(N); WriteLn('Введіть початкове значення
аргументу');
Write('X0='); ReadLn(x); WriteLn('Введіть кінцеве значення аргументу');
Write('X_MAX='); ReadLn(xm); WriteLn('Задайте погрішність обчислень');
Write('E='); ReadLn(e1); WriteLn('Введіть початкове значення');
for j:=1 to N do begin Write('Y0(',j')='); ReadLn(w[j]); y[j]:=w[j]; end;
WriteLn('Введіть початкове значення кроку інтеграції'); Write('h=');
ReadLn(h); repeat e3:=0; V1; d1:=0; for j:=1 to N do begin a[j]:=F[j]*h;
y[j]:=w[j]+(a[j]/3); end; x:=x+(h/3); V1; for j:=1 to N do begin
y[j]:=w[j]+((a[j]+F[j]*h)/6); end; V1; for j:=1 to N do begin з[j]:=F[j]*h;
y[j]:=w[j]+(a[j]/8)+0.375*c[j]; end; x:=x+(h/6); V1; for j:=1 to N do begin
d[j]:=F[j]*h; y[j]:=w[j]+(a[j]/2)-1.5*c[j]+2*d[j]; end; x:=x+h/2; V1; for
j:=1 to N do begin e[j]:=F[j]*h; y[j]:=w[j]+(a[j]+4*d[j]+e[j])/6; e2:=(abs(-
```

```
2*a[j]+9*c[j]-8*d[j]+e[j])) /30;  if e2<=e1 then  begin  if e2<(e1/20) then
begin  d1:=d1+1;  end;
end  else  begin  e3:=1;  end;  end; if e3=0 then  begin  if d1=N then  begin
h:=h+h;  end;  Write('X=',x:5:2);  for j:=1 to N do  begin Write('
Y(',j')=',y[j]:12:9);
w[j]:=y[j];  end;  end else  begin  x:=x-h;  for j:=1 to N do  begin  y[j]:=w[j];
end;  h:=h/2;  end;  WriteLn;  until x>xm;END.
```

PROGRAM DIF_UR_SIS_M_T;

```
{ Вирішення системи лінійних диференціальних рівнянь
методом трапецій} USES  Crt;  var  N,j: integer;  x,h,xm: real;
y:array[1..50] of real;  w:array[1..50] of real; k:array[1..50] of real;
F:array[1..50] of real; Procedure V;
 { Представляємо систему диференціальних рівнянь dyi/dx=fi(x,yi)
  ввиде F(i)=fi(x,y)}  Begin  F[1]:=y[2];  F[2]:=(((y[1]/x)-y[2])/x)-y[1];  end;
begin  ClrScr;
WriteLn('Задайте число диференціальних рівнянь першого порядку');
Write('N=');  ReadLn(N);  WriteLn('Введіть початкове значення
аргументу');
Write('X0=');  ReadLn(x);  WriteLn('Введіть кінцеве значення аргументу');
Write('X_MAX=');  ReadLn(xm);  WriteLn('Введіть початкове значення');
for j:=1 to N do  begin  Write('Y0(',j')=');  ReadLn(w[j]);  y[j]:=w[j];
end;  WriteLn('Введіть початкове значення кроку інтеграції');
Write('h=');  ReadLn(h);  repeat  V;  for j:=1 to N do  begin
до[j]:=h*F[j]; y[j]:=w[j]+k[j];  end;  x:=x+h;  Write('X=',x:5:2);  V;  for j:=1 to
N do
begin  y[j]:=w[j]+((до[j]+h*F[j])/2); Write('  Y(',j')=',y[j]:12:9);  w[j]:=y[j];
end;
WriteLn;  until x>xm;END.
```

PROGRAM DIF_UR_SIS_M_RK4;

```
{ Вирішення системи лінійних диференціальних рівнянь
методом Рунге-Кутта четвертого порядку}  USES Crt;  var  N,j: integer;
x,h,xm,v: real;  balon: text; y:array[1..50] of real;  w:array[1..50] of real;
k:array[1..50] of real;  F:array[1..50] of real;  a:array[1..50] of real; Procedure
V1;
{ Представляємо систему диференціальних рівнянь dyi/dx=fi(x,yi)
ввиде F(i)=fi(x,y)}  Begin  F[1]:=(25/100000)*((sin(y[2])
/cos(y[2]))/cos(y[3]))-7*x; F[2]:=(1/y[1])*((sin(y[2])
/cos(y[2]))*(7*x+x*x*(sin(y[3]) /cos(y[3])))+((25/100000)/cos(y[3])));
  F[3]:=((7*x*sin(y[3]) -(x*x) -
0.52*cos(y[2]))/(y[1]*cos(y[2])*cos(y[2])*cos(y[3])))-((sin(y[3]) /cos(y[3]))/x);
F[4]:=(sin(y[3]) /cos(y[3]))/x;
  F[5]:=(sin(y[2]) /cos(y[2]))/cos(y[3]);  end;begin       ClrScr;
Assign(balon,'C:\balon.txt');  Rewrite(balon);
WriteLn('Задайте число диференціальних рівнянь першого порядку');
Write('N=');  ReadLn(N); WriteLn('Введіть початкове значення аргументу');
Write('X0='); ReadLn(x);  WriteLn('Введіть кінцеве значення аргументу');
Write('X_MAX=');  ReadLn(xm);   WriteLn('Введіть початкове значення');
for j:=1 to N do  begin Write('Y0(',j')='); ReadLn(w[j]);  y[j]:=w[j];   end;
WriteLn('Введіть початкове значення кроку інтеграції'); Write('h=');
ReadLn(h);  repeat  V1;  for j:=1 to N do  begin  v:=h*F[j]; до[j]:=v;
y[j]:=w[j]+(v/2);
end;  x:=x+(h/2);  V1;  for j:=1 to N do  begin v:=h*F[j]; до[j]:=k[j]+2*v;
y[j]:=w[j]+(v/2);  end;  V1;  for j:=1 to N do  begin  v:=h*F[j]; до[j]:=k[j]+2*v;
y[j]:=w[j]+v;  end;  x:=x+(h/2);  Write('X=',x:6:3); Write(balon,'X=',x:6:3);
V1;  for j:=1 to N do  begin  y[j]:=w[j]+((до[j]+(h*F[j]))/6);  Write(balon',
Y(',j')=',y[j]:10:5); Write('   Y(',j')=',y[j]:10:5); w[j]:=y[j];  end; WriteLn(balon);
WriteLn;  until x>xm; Close(balon);END.
```

```
PROGRAM IN_M_PR;
{ Програма для обчислення певного інтеграла методом
прямокутників}
uses Crt; var a,b,m,h,x,y,s,io:real; i:integer; {Завдання підінтегральній
функції} PROCEDURE V; begin y:=sqrt(2*x+1) end;begin ClrScr;
WriteLn('Введення верхньої і нижньої меж інтеграції a,b');
Write('b='); Read(b); Write('a='); Read(a); WriteLn('Задайте число ділянок
розбиття'); Write('m='); Read(m); h:=(b-a) /m; x:=a; s:=0;i:=0; repeat
begin V; s:=s+y; x:=x+h; end; i:=i+1; until i>m-1; io:=s*h;
Write('I=',io:16:8);
END.
PROGRAM IN_M_SIM;
{ Програма для обчислення певного інтеграла методом
Сімпсона} uses Crt; var a,b,h,x,y,io,s,ia,ib:real; i,m:integer;
{Завдання підінтегральній функції} PROCEDURE V; begin
y:=sqrt(2*x+1)
end;begin ClrScr; WriteLn('Введення верхньої і нижньої меж
інтеграції a,b'); Write('b='); Read(b); Write('a='); Read(a); WriteLn('Задайте
число ділянок розбиття'); Write('m='); Read(m); h:=(b-a) /m; x:=a; V;
ia:=y*h/3; x:=b; V; ib:=y*h/3; s:=0;x:=a;s:=0;i:=1;
repeat begin x:=x+h; V; if Odd(i) then begin s:=s+4*y; end else begin
s:=s+2*y; end; end; i:=i+1 until i>m-1; io:=s*(h/3)+ia+ib;
Write('I=',io:16:8);
END.
PROGRAM IN_M_TR;
{ Програма для обчислення певного інтеграла методом
трапецій}
```

```
uses Crt; var a,b,h,x,y,io,s,ia,ib:real; i,m:integer; {Завдання
підінтегральній функції} PROCEDURE V; begin y:=sqrt(2*x+1)
end;begin ClrScr;
WriteLn('Введення верхньої і нижньої меж інтеграції a,b'); Write('b=');
Read(b); Write('a='); Read(a);
WriteLn('Задайте число ділянок розбиття'); Write('m='); Read(m); h:=(b-
a) /m; x:=a; V; ia:=y*h/2; x:=b; V; ib:=y*h/2; s:=0;x:=a;s:=0;i:=1; repeat
begin x:=x+h; V; s:=s+y; end; i:=i+1 until i>m-1; io:=s*h+ia+ib;
Write('I=',io:16:8);END.
```

```
PROGRAM AP_M_NK;
{ Апроксимація і інтерполяція результатів досліджень поліномом по
методу найменших квадратів з можливістю автоматичного вибору ступеня
полінома}
{Поліном має вигляд y(x)=a0+a1*x+a2*x^2+a3*x^3+...+an*x^n} uses Crt;
var
i,H,VV,N,j,k,N1,k1,j1,M,L,M2:integer; E1,R,F,U,S,E,p,Q1:real; X:array[1..50]
of real; Y:array[1..50] of real; Z:array[1..50] of real; B:array[1..50] of real;
G:array[1..50] of real; A:array[1..50,1..50] of real; C:array[1..50,1..50] of real;
D:array[1..100] of real;label M1;PROCEDURE V1V1; begin N:=N+1;
for j:=0 to 2*N-1 do begin if j>N then begin D[j]:=0; end else begin
B[j]:=0;
D[j]:=0; end; end; for i:=1 to H do begin R:=Y[i]; F:=1; for j:=1 to 2*N-1 do
begin if j>N then begin D[j]:=D[j]+F; F:=F*X[i]; end else begin
B[j]:=B[j]+R; R:=R*X[i]; D[j]:=D[j]+F; F:=F*X[i]; end; end; end; for i:=1 to
N do
begin k:=i; for j:=1 to N do begin A[i,j]:=D[k]; k:=k+1; end; end; N1:=N-1;
for k:=1 to N1 do begin if abs(A[до, до]) >0 then begin end else begin
k1:=k+1;
```

```
for M:=k1 to N do  begin  if abs(A[M,k])=0 then  begin  end else begin for L:=1
to N do   begin  U:=A[к,L]; A[к,L]:=A[M,L];  A[M,L]:=U;  end;  end;  end;
U:=B[k];
B[k]:=B[M]; B[M]:=U;end;  G[k]:=B[k]/A[до, до]; k1:=k+1; for i:=k1 to N do
begin  B[i]:=B[i]-A[i,k]*G[k];   for j1:=k to N do  begin  j:=N-j1+k;
C[к,j]:=A[к,j]/A[до, до]; A[i,j]:=A[i,j]-A[i,k]*C[к,j];    end;   end;    end;
M:=N; Z[M]:=B[M]/A[M,M];   repeat   M:=M-1;    S:=0;    for L:=M to N1 do
begin   S:=S+C[M,L+1]*Z[L+1];   end;  Z[M]:=G[M]-S;   until M<=1;  E:=0;
for i:=1 to H do   begin S:=Y[i]; R:=1;  for j:=1 to N do  begin  S:=S-R*Z[j];
R:=R*X[i]; end;  E:=E+S*S;  end; E:=sqrt(E/H); end;PROCEDURE V1V2;
begin  WriteLn('Ступінь полінома N= ',N-1:4);  WriteLn('Погрішність
результатів E=',e); WriteLn('Коефіцієнти полінома');  for i:=1 to N do
begin  WriteLn('A(',i-1')=',Z[i]:14:5);  end; end;PROCEDURE V1V3; begin
WriteLn('Ступінь полінома N= ',N-1:4);  WriteLn('Погрішність результатів
E=',e);
WriteLn('Якщо E більше заданою, збільшуємо N і повторюємо
обчислення');
WriteLn('Коефіцієнти полінома'); for i:=1 to N do  begin  WriteLn('A(',i-
1')=',Z[i]:14:5); end;  end;begin  ClrScr;
Write('Задайте число вузлів інтерполяції або апроксимації'); Write(' N=');
Read(H);  WriteLn('Введення значень X(i) і Y(i) для кожного вузла ');
for i:=1 to H do  begin  Write('X(',i')=');  Read(X[i]);
Write('Y(',i')=');Read(Y[i]);
end;  N:=1; WriteLn('Введіть 0 - якщо задана погрішність апроксимації');
WriteLn('        1 - якщо задається ступінь полінома');  Read(VV); if VV=0
then
begin WriteLn('Задайте середньоквадратичну погрішність');Write('E1=');
Read(E1); end  else  begin WriteLn('Введіть ступінь полінома M<N');
```

```
Write('M='); Read(N); end; V1V1; if VV=1 then begin V1V2; end else
begin
M1: if E<=E1 then begin V1V3; end else begin if N<H then begin V1V1;
goto M1; end else begin V1V3; end; end; end;
WriteLn('Перевірочне обчислення величини Y(X)'); Write('Введіть
значення X='); Read(p); Q1:=0; for i:=N downto 2 do begin
Q1:=(Q1+Z[i])*p;
end; Q1:=Q1+Z[1]; Write('Значення Y(X)=',Q1:14:5);END.
```

Програма для числельного інтегрування системи диференційних рівнянь з використанням метода Рунге-Кутта-Мерсона з автоматичним обранням крока інтегрування

```
unit SDU1; interface uses Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls, jpeg, ExtCtrls;
type TfrmSDU1 = class(TForm) lbl1SDU1: TLabel; lbl2SDU1: TLabel;
lbl4SDU1: TLabel; lbl5SDU1: TLabel; lbl6SDU1: TLabel;
lbl7SDU1: TLabel; btn1SDU1: TButton; Image1: TImage; Label1: TLabel;
Label2: TLabel; Label3: TLabel; Label4: TLabel; Label5: TLabel;
procedure btn1SDU1Click(Sender: TObject); private { Private declarations }
public { Public declarations } end; var frmSDU1: TfrmSDU1;
implementation uses SDU2; {$R *.dfm}
procedure TfrmSDU1.btn1SDU1Click(Sender: TObject); begin frmSDU1.Hide;
frmSDU2.Show; end; unit SDU2; interface uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,Math,Chart, ExtCtrls, TeeProcs, TeEngine, Series;
Type TfrmSDU2 = class(TForm) btn1SDU2: TButton; lbl1SDU2: TLabel;
lbl2SDU2: TLabel; lbl3SDU2: TLabel; lbl4SDU2: TLabel; lbl5SDU2:
TLabel;
lbl6SDU2: TLabel; lbl7SDU2: TLabel; lbl8SDU2: TLabel; Memo1: TMemo;
```

```
edt1SDU2: TEdit; edt2SDU2: TEdit; edt3SDU2: TEdit; edt4SDU2: TEdit;
edt5SDU2: TEdit; mem1SDU2: TMemo; btn2SDU2: TButton;
btn3SDU2: TButton; Chart1: TChart; Button1: TButton; Series1: TLineSeries;
Series2: TLineSeries; procedure btn1SDU2Click(Sender: TObject);
procedure btn3SDU2Click(Sender: TObject);
procedure btn2SDU2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);   private { Private declarations }
public { Public declarations } end; type mm=array[1..50]of Real;
var frmSDU2: TfrmSDU2;   N,j,d1,code,i,t,t1:Integer;
x,h,xm,e1,e2,e3,b1,b2,b3,b4:Real; y,w,c,F,a,d,e:mm;
 Y0,xx,yy:array[1..50]of String;     ii,xxx,jj:String;     aa,aaa,aaa1:array[1..200]of
Real; implementation uses SDU1,Synt,UErrors; {$R *.dfm}
procedure V1(var F:mm; X:Real;y:mm);   begin SetData('X',X);
SetData('Y[1]',Y[1]); SetData('Y[2]',Y[2]);      SetData('Y[3]',Y[3]);
SetData('Y[4]',Y[4]); SetData('Y[5]',Y[5]);      SetData('Y[6]',Y[6]);
SetData('Y[7]',Y[7]); SetData('Y[8]',Y[8]);      SetData('Y[9]',Y[9]);
SetData('Y[10]',Y[10]); SetData('Y[11]',Y[11]); SetData('Y[12]',Y[12]);
SetData('Y[13]',Y[13]); SetData('Y[14]',Y[14]); SetData('Y[15]',Y[15]);
SetData('Y[16]',Y[16]); SetData('Y[17]',Y[17]); SetData('Y[18]',Y[18]);
SetData('Y[19]',Y[19]); SetData('Y[20]',Y[20]); SetData('Y[21]',Y[21]);
SetData('Y[22]',Y[22]); SetData('Y[23]',Y[23]); SetData('Y[24]',Y[24]);
for i:=1 to N do begin Calculate(F[i]); end; GetData('F[1]',F[1]);
GetData('F[2]',F[2]); GetData('F[3]',F[3]); GetData('F[4]',F[4]);
GetData('F[5]',F[5]); GetData('F[6]',F[6]); GetData('F[7]',F[7]);
GetData('F[8]',F[8]); GetData('F[9]',F[9]); GetData('F[10]',F[10]);
GetData('F[11]',F[11]); GetData('F[12]',F[12]); GetData('F[13]',F[13]);
GetData('F[14]',F[14]); GetData('F[15]',F[15]); GetData('F[16]',F[16]);
GetData('F[17]',F[17]); GetData('F[18]',F[18]); GetData('F[19]',F[19]);
GetData('F[20]',F[20]); GetData('F[21]',F[21]); GetData('F[22]',F[22]);
```

```
GetData('F[23]',F[23]); GetData('F[24]',F[24]); end;
procedure TfrmSDU2.btn1SDU2Click(Sender: TObject); begin
frmSDU1.Close; end; procedure TfrmSDU2.btn3SDU2Click(Sender: TObject);
begin edt1SDU2.Clear; edt2SDU2.Clear; edt3SDU2.Clear;
edt4SDU2.Clear; edt5SDU2.Clear; Memo1.Clear; mem1SDU2.Clear;
end; procedure TfrmSDU2.btn2SDU2Click(Sender: TObject); begin
Val(edt1SDU2.Text,N,code); Val(edt2SDU2.Text,x,code);
Val(edt3SDU2.Text,xm,code); Val(edt4SDU2.Text,e1,code);
Val(edt5SDU2.Text,h,code); for i:=1 to N do begin ii:= format('%2.0d',[i]);
mem1SDU2.Lines.Add(y'); Y0[i]:=InputBox( Y0('+ii+')=','');
Val(Y0[i],w[i],code); y[i]:=w[i]; xx[i]:=format('%12.5f',[w[i]]);
mem1SDU2.Lines.Add('Y0('+ii+')='+xx[i]); end;
if (FErrors <> nil) then FErrors.Close; if not CreatePZ(Memo1.Text)
then begin Application.CreateForm(TFErrors, FErrors);
FErrors.LBErrors.Items.Assign(ErrorList); FErrors.Show; exit; end;
t:=0; repeat t:=t+1; e3:=0; V1(F,X,Y); d1:=0; for j:=1 to N do begin
a[j]:=F[j]*h; y[j]:=w[j]+(a[j]/3); end; x:=x+(h/3); V1(F,X,Y);
for j:=1 to N do begin y[j]:=w[j]+((a[j]+F[j]*h)/6); end; V1(F,X,Y);
for j:=1 to N do begin c[j]:=F[j]*h; y[j]:=w[j]+(a[j]/8)+0.375*c[j]; end;
x:=x+(h/6); V1(F,X,Y); for j:=1 to N do begin d[j]:=F[j]*h;
y[j]:=w[j]+(a[j]/2)-1.5*c[j]+2*d[j]; end; x:=x+h/2; V1(F,X,Y);
for j:=1 to N do begin e[j]:=F[j]*h; y[j]:=w[j]+(a[j]+4*d[j]+e[j])/6;
e2:=(abs(-2*a[j]+9*c[j]-8*d[j]+e[j]))/30; if e2<=e1 then begin if e2<(e1/20)
then
begin d1:=d1+1; end; end else begin e3:=1; end; end; if e3=0 then
begin if d1=N then begin h:=h+h; end; aa[t]:=x; aaa[t]:=y[1]; aaa1[t]:=y[2];
xxx:=Format('%17.8f',[x]); mem1SDU2.Lines.Add('X='+xxx);
for j:=1 to N do begin jj:=format('%2.0d',[j]); yy[j]:=format('%17.8f',[y[j]]);
mem1SDU2.Lines.Add('Y['+jj+']='+yy[j]); w[j]:=y[j]; end; end else begin
```

```
x:=x-h; for j:=1 to N do begin y[j]:=w[j]; end; h:=h/2; end; until x>xm;
t1:=t; end; procedure TfrmSDU2.Button1Click(Sender: TObject); begin
with Series1 do begin for t:=1 to t1-1 do begin
Series1.AddXY(aa[t],aaa[t],",clRed); {Series2.AddXY(aa[t],aaa1[t],",clRed);}
end; end; with Series2 do begin for t:=1 to t1-1 do begin
Series2.AddXY(aa[t],aaa1[t],",clBlue); end; end; end; unit Synt; interface
uses classes; type TData = record Name: string;    Data:real;  end;
var NConst: integer = 100; ErrorList: TStringList; PZ: array of integer;
DataList: array of TData; const MConst = 2;
procedure SyntItem(S:string; First:boolean=false; Pos:Integer=1);
function CreatePZ(S:string):boolean; function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean;
function GetData(Name:string; var Data:real):boolean;
implementation uses Sysutils, Math, Dialogs; type
TType = (None, Number, Divider, Ident, Func, Part, All); TSynt = record
mode: TType;        Number:real    Ident:string;          Error:boolean;
Pos1,Pos2:integer;  end; const SetNum: set of char=['0'..'9', ',';
SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^', '{', ' ', #13];
SetChar: set of char=['a'..'z','A'..'Z','_']; NFunc = 9;
Functions: array[1..NFunc] of string =
('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos');
Var SItem: TSynt; TrStack: array of char; ConstList: array of real;
Position: Integer;
procedure SyntItem(S:string;First:boolean=false;Pos:Integer=1);
var i:integer; begin if (S = '') then begin SItem.mode := All; exit; end;
if(First) then Position := Pos; repeat if (S[Position] = '{') then begin
repeat Inc(Position) until (Position >= Length(S)) or (S[Position] = '}');
Inc(Position);    end; if(Position <= Length(S)) then while ((S[Position] = ' ')or
(S[Position] = #13)or (S[Position] = #10)or (S[Position] = #0)) do Inc(Position);
```

```
until (S[Position] <> '{'); SItem.Error:=false; SItem.Pos1:=Position;
if(Position > Length(S)) then begin SItem.mode := All; exit;  end;
SItem.Ident := S[Position]; if (S[Position] in SetChar) then SItem.mode := Ident
else if (S[Position] in SetNum) then SItem.mode := Number
else if (S[Position] in SetDiv) then begin if (S[Position] <> ';')
then SItem.mode := Divider else SItem.mode := Part; Inc(Position); exit;
end else begin SItem.mode := None; Inc(Position);   exit; end; repeat
Inc(Position); if (SItem.mode = Number)and
((S[Position] = '-')or(S[Position] = '+'))and (UpCase(S[Position-1])='E')
then SItem.Ident := SItem.Ident + S[Position]
else if ((Position > Length(S))or(S[Position] in SetDiv)) then begin
if(SItem.mode = Number) then try SItem.Number := StrToFloat(SItem.Ident)
except on EConvertError do SItem.Error := true; end; for i:=1 to NFunc do
if (LowerCase(SItem.Ident) = Functions[i]) then begin SItem.mode:=Func;
SItem.Number:=i; break; end; SItem.Pos2:=Position-1; exit; end
else SItem.Ident := SItem.Ident + S[Position]; until false; end; procedure
ClearPZ;
begin ErrorList.Clear; SetLength(ConstList,0); SetLength(DataList,MConst);
SetLength(PZ,0); end; function CreatePZ(S:string):boolean; var
lend:boolean; i:integer; Assign:boolean; Adress: integer; OldMode: TType;
OldS: charprocedure code; begin SetLength(PZ,High(PZ)+2);
case TrStack[High(TrStack)] of   '+': PZ[High(PZ)] := -1;   '-': PZ[High(PZ)] := -
2;
'*': PZ[High(PZ)] := -3;   '/': PZ[High(PZ)] := -4; '^': PZ[High(PZ)] := -5;
'M': PZ[High(PZ)] := -6; end; end; procedure proc1; begin
SetLength(TrStack,High(TrStack)+2);          TrStack[High(TrStack)]          :=
SItem.Ident[1];
end; procedure proc2; begin code; TrStack[High(TrStack)] := SItem.Ident[1];
end; procedure proc3; begin code; SetLength(TrStack,High(TrStack));
```

```
lend:=false; end; procedure proc4; begin SetLength(TrStack,High(TrStack));
end; procedure proc5; SetLength(TrStack,High(TrStack)+2);
TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
end; procedure proc6; SetLength(PZ,High(PZ)+2);
PZ[High(PZ)] := -Ord(TrStack[High(TrStack)])+27;
SetLength(TrStack,High(TrStack)); end; begin ClearPZ; SetLength(TrStack,1);
TrStack[0] := '0'; OldMode := None; OldS := ' '; Assign := true; Adress := 0;
SyntItem(S,true);  if (SItem.mode = All) then begin ErrorList.Add();
Result := false; exit; end; repeat if ((OldMode = Func)and(SItem.Ident[1] <> '('))
then ErrorList.Add('+IntToStr(SItem.Pos1)); case SItem.mode of
Number: begin if((OldMode <> Divider)and(OldMode <> None)and
(OldMode <> Part)) then ErrorList.Add('+IntToStr(SItem.Pos1)+' );
if (SItem.Error) then ErrorList.Add('+IntToStr(SItem.Pos1)+
' - '+IntToStr(SItem.Pos2)) else begin SetLength(ConstList,High(ConstList)+2);
ConstList[High(ConstList)] := SItem.Number; SetLength(PZ,High(PZ)+2);
PZ[High(PZ)] := High(ConstList); end; Assign:=false; end; Ident:  begin
if((OldMode <> Divider)and(OldMode <> None)and (OldMode <> Part))
then ErrorList.Add('Â ïîçèöèè '+IntToStr(SItem.Pos1)+' );
for i:=0 to High(DataList) do begin
if (UpperCase(SItem.Ident) = DataList[i].Name)
then begin SetLength(PZ,High(PZ)+2); PZ[High(PZ)] := NConst+i; break;
end; if(i = High(DataList)) then begin SetLength(DataList,High(DataList)+2);
DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
DataList[High(DataList)].Data:=0; SetLength(PZ,High(PZ)+2);
PZ[High(PZ)] := NConst+High(DataList); end; end; end;
All,Part:begin repeat lend:=true; case TrStack[High(TrStack)] of
'0': begin if (Adress <> 0) then begin SetLength(PZ,High(PZ)+3);
PZ[High(PZ)-1] := -7; PZ[High(PZ)] := Adress; Adress := 0; end; break; end;
'(': ErrorList.Add(); else proc3; end; until lend; if (ErrorList.Count = 0)
```

```
then Result:=true else Result:=false; if (SItem.mode = All) then exit
else begin Assign := true; SItem.mode := None; end end; Divider:begin
if((OldMode = Divider)and ((SItem.Ident[1]<>'=')and (SItem.Ident[1]<>'(')and
(SItem.Ident[1] <> ')'))and ((OldS <> '(')and (OldS <> ')')and (OldS <> '=')))
then begin ErrorList.Add(' '+IntToStr(SItem.Pos1)+': ); break; end; repeat
lend:=true; case SItem.Ident[1] of '=': if Assign and (OldMode = Ident)
then begin Adress := PZ[High(PZ)]; SetLength(PZ,High(PZ));
SItem.mode := None; end else ErrorList.Add('Îîçèöèÿ '+IntToStr(SItem.Pos1)+
': ); '(': if(OldMode = Ident) or (OldMode = Number)
then ErrorList.Add('+IntToStr(SItem.Pos1)) else  proc1; '+','-','M': begin
if((OldMode = None)or(OldS = '(')) then if (SItem.Ident[1] = '+') //
then break else SItem.Ident[1] := 'M'; case TrStack[High(TrStack)] of
'0','(': proc1; '+','-','M': proc2; '*','/','^': proc3; end; end; '*','/': if OldS = '('
then ErrorList.Add('+IntToStr(SItem.Pos1)) else case TrStack[High(TrStack)]
of
'0','(','+','-','M': proc1; '*','/': proc2; '^': proc3; end; '^': if OldS = '('
then ErrorList.Add('+IntToStr(SItem.Pos1)) else case TrStack[High(TrStack)]
of
'0','(','+','-','*','/','M': proc1; '^': proc2; end; ')':case TrStack[High(TrStack)] of
'0': ErrorList.Add(); '(': begin proc4; if (Ord(TrStack[High(TrStack)]) > 127)
then proc6; end; '+','-','*','/','^','M': proc3; end; end; until lend; Assign:=false;
end; Func:   begin repeat lend:=true; proc5 until lend; Assign:=false;
end; None:   ErrorList.Add('+IntToStr(SItem.Pos1)); end;
OldMode := SItem.mode; OldS := SItem.Ident[1]; SyntItem(S); until false;
if(ErrorList.Count = 0) then Result := true else Result := false; end;
function Calculate(var R:real):boolean; var Stack: array of real; i:integer;
begin for i:=0 to High(PZ) do begin if (i > 0) then
if (PZ[i-1] = -7) and (i < High(PZ)) then Continue; if PZ[i] < -100 then begin
try case -PZ[i]-100 of 1: Stack[High(Stack)]:=Exp(Stack[High(Stack)]);
```

```
2: Stack[High(Stack)]:=Sin(Stack[High(Stack)]);
3: Stack[High(Stack)]:=Cos(Stack[High(Stack)]);
4: Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);
5: Stack[High(Stack)]:=Abs(Stack[High(Stack)]);
6: Stack[High(Stack)]:=Ln(Stack[High(Stack)]);
7: Stack[High(Stack)]:=Tan(Stack[High(Stack)]);
8: Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);
9: Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]); end except Result := false;
exit; end; if(FloatToStr(Stack[High(Stack)]) = 'NAN') or
(FloatToStr(Stack[High(Stack)]) = 'INF') or
(FloatToStr(Stack[High(Stack)]) = '-INF') then begin Result := false; exit;
End end else if PZ[i] < 0 then begin try case -PZ[i] of 1: Stack[High(Stack)-1]:=
Stack[High(Stack)-1]+Stack[High(Stack)]; 2: Stack[High(Stack)-1]:=
Stack[High(Stack)-1]-Stack[High(Stack)]; 3: Stack[High(Stack)-1]:=
Stack[High(Stack)-1]*Stack[High(Stack)]; 4: Stack[High(Stack)-1]:=
Stack[High(Stack)-1]/Stack[High(Stack)]; 5: Stack[High(Stack)-1]:=
Power(Stack[High(Stack)-1],Stack[High(Stack)]);
6: Stack[High(Stack)]:= -Stack[High(Stack)];
7: DataList[PZ[i+1]-NConst].Data := Stack[High(Stack)]; end; except
Result := false;exit; end; if (PZ[i] <> -6) then SetLength(Stack,High(Stack));
End else begin SetLength(Stack,High(Stack)+2); if (PZ[i] < NConst)
then Stack[High(Stack)]:=ConstList[PZ[i]]
else Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data; end; end;
Result := true; R :=Stack[High(Stack)]; end;
function SetData(Name:string; Data:real):boolean; var i:integer; begin
for i:=MConst to High(DataList) do if (UpperCase(Name) = DataList[i].Name)
then  begin DataList[i].Data := Data; Result:=true; exit; end; Result := false;
end; function GetData(Name:string; var Data:real):boolean; var i:integer;
begin for i:=0 to High(DataList) do if (UpperCase(Name) = DataList[i].Name)
```

then begin Data := DataList[i].Data; Result:=true; exit; end; Result := false; end; initialization SetLength(DataList,MConst); DataList[0].Name:='PI'; DataList[0].Data:=Pi; DataList[1].Name:='E'; DataList[1].Data:=2.71828183; ErrorList := TStringList.Create; finalization ErrorList.Free; unit UErrors; Interface uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, StdCtrls; type TFErrors = class(TForm) LBErrors: TListBox; private { Private declarations } public { Public declarations } end; var FErrors: TFErrors; implementation {$R *.dfm}

Програма для апроксімації та інтерполяції за методом найменьших квадратів з можливістю автоматичного обрання ступеня полінома

unit APIN1; interface uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, jpeg, ExtCtrls, StdCtrls; type TfrmAPIN1 = class(TForm) lbl1APIN1: TLabel; lbl2APIN1: TLabel; lbl3APIN1: TLabel; lbl4APIN1: TLabel; lbl5APIN1: TLabel; lbl6APIN1: TLabel; lbl7APIN1: TLabel; btn1APIN1: TButton; img1APIN1: TImage; Label1: TLabel; procedure btn1APIN1Click(Sender: TObject); private { Private declarations } public { Public declarations } end; var frmAPIN1: TfrmAPIN1; implementation uses APIN2; {$R *.dfm} procedure TfrmAPIN1.btn1APIN1Click(Sender: TObject); begin frmAPIN1.Hide; frmAPIN2.Show; end; unit APIN2; interface uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,Math;type TfrmAPIN2 = class(TForm) lbl1APIN2: TLabel; lbl2APIN2: TLabel;lbl3APIN2: TLabel; edt1APIN2: TEdit; edt2APIN2: TEdit;
lbl4APIN2: TLabel; mem1APIN2: TMemo; btn1APIN2: TButton; btn2APIN2: TButton; btn3APIN2: TButton; lbl5APIN2: TLabel; edt3APIN2: TEdit; procedure btn3APIN2Click(Sender: TObject);

```
procedure btn2APIN2Click(Sender: TObject);
procedure btn1APIN2Click(Sender: TObject);private { Private declarations }
public { Public declarations } end; var frmAPIN2: TfrmAPIN2;
i,H,VV,N,j,k,N1,k1,j1,M,L,M2,code:Integer; E1,R,F,U,S,E,p,Q1:Real;
X:array[1..50]of Real; Y:array[1..50]of Real; Z:array[1..50]of Real;
B:array[1..50]of Real; G:array[1..50]of Real; A:array[1..50,1..50]of Real;
C:array[1..50,1..50]of Real; D:array[1..100]of Real;
vv7,EE1,EE2,NN1,EEE,ikpl,kpl:String; aa:array[1..50]of String;
aa1:array[1..50]of String; implementation uses APIN1; {$R *.dfm}
PROCEDURE V1V1; begin N:=N+1; for j:=0 to 2*N-1 do begin if j>N then
Begin D[j]:=0; end else begin B[j]:=0; D[j]:=0; end; end; for i:=1 to H do
Begin R:=Y[i]; F:=1; for j:=1 to 2*N-1 do begin if j>N then begin
D[j]:=D[j]+F; F:=F*X[i]; end else begin B[j]:=B[j]+R; R:=R*X[i];
D[j]:=D[j]+F; F:=F*X[i]; end; end; end; for i:=1 to N do begin k:=i;
for j:=1 to N do begin A[i,j]:=D[k]; k:=k+1; end; end; N1:=N-1; for k:=1 to N1
do
begin if abs(A[k,k])>0 then begin end else begin k1:=k+1; for M:=k1 to N do
begin if abs(A[M,k])=0 then begin end else begin for L:=1 to N do begin
U:=A[k,L]; A[k,L]:=A[M,L]; A[M,L]:=U; end; end; end; U:=B[k]; B[k]:=B[M];
B[M]:=U; end; G[k]:=B[k]/A[k,k]; k1:=k+1; for i:=k1 to N do begin
B[i]:=B[i]-A[i,k]*G[k]; for j1:=k to N do begin j:=N-j1+k;
C[k,j]:=A[k,j]/A[k,k];
A[i,j]:=A[i,j]-A[i,k]*C[k,j]; end; end; end; M:=N; Z[M]:=B[M]/A[M,M];
Repeat M:=M-1; S:=0; for L:=M to N1 do begin S:=S+C[M,L+1]*Z[L+1];
end; Z[M]:=G[M]-S; until M<=1; E:=0; for i:=1 to H do begin S:=Y[i]; R:=1;
for j:=1 to N do begin S:=S-R*Z[j]; R:=R*X[i]; end; E:=E+S*S; end;
E:=sqrt(E/H); end;procedure TfrmAPIN2.btn3APIN2Click(Sender: TObject);
Begin frmAPIN1.Close; end;
procedure TfrmAPIN2.btn2APIN2Click(Sender: TObject); begin
```

```
edt1APIN2.Clear; edt2APIN2.Clear; edt3APIN2.Clear; mem1APIN2.Clear;
end; procedure TfrmAPIN2.btn1APIN2Click(Sender: TObject); label M1;
begin Val(edt1APIN2.Text,H,code); for i:=1 to H do begin
vv7:=format('%3.0d',[i]);
aa[i]:=InputBox('Введение значений',' X['+vv7+']=',''); Val(aa[i],X[i],code);
mem1APIN2.Lines.Add('X('+vv7+')='+aa[i]);
aa1[i]:=InputBox('Введение значений',' Y['+vv7+']=',''); Val(aa1[i],Y[i],code);
mem1APIN2.Lines.Add('Y('+vv7+')='+aa1[i]); end; N:=1;
Val(edt2APIN2.Text,VV,code); if VV=0 then  begin
EE1:=InputBox('Среднеквадратичная погрешность','Задайте
среднеквадратичную погрешность',''); Val(EE1,E1,code);
edt3APIN2.Text:=EE1; end else begin
EE2:=InputBox('Степень полинома','Введите степень M полинома
(M<N)','');
Val(EE2,N,code); end; V1V1; if VV=1 then begin NN1:=format('%3.0d',[N-1]);
mem1APIN2.Lines.Add('Степень полинома N='+NN1);
EEE:=format('%9.5f',[E]);
mem1APIN2.Lines.Add('Погрешность результатов E='+EEE);
mem1APIN2.Lines.Add('Коэффициенты полинома'); for i:=1 to N do
begin ikpl:=format('%2.0d',[i-1]); kpl:=format('%14.5f',[Z[i]]);
mem1APIN2.Lines.Add('A('+ikpl+')='+kpl); end; end else begin M1:
if E<=E1 then begin NN1:=format('%3.0d',[N-1]);
mem1APIN2.Lines.Add('Степень полинома N='+NN1);
EEE:=format('%9.5f',[E]);
mem1APIN2.Lines.Add('Погрешность результатов E='+EEE);
mem1APIN2.Lines.Add('Если E больше заданной, увеличиваем N и
повторяем вычисления'); mem1APIN2.Lines.Add('Коэффициенты
полинома');
for i:=1 to N do begin ikpl:=format('%2.0d',[i-1]); kpl:=format('%14.5f',[Z[i]]);
```

```
mem1APIN2.Lines.Add('A('+ikpl+')='+kpl); end; end else begin if N<H then
begin V1V1; goto M1; end else begin NN1:=format('%3.0d',[N-1]);
mem1APIN2.Lines.Add('Степень полинома N='+NN1);
EEE:=format('%9.5f',[E]);
mem1APIN2.Lines.Add('Погрешность результатов E='+EEE);
mem1APIN2.Lines.Add('Если E больше заданной, увеличиваем N и
повторяем вычисления'); mem1APIN2.Lines.Add('Коэффициенты
полинома');
for i:=1 to N do begin ikpl:=format('%2.0d',[i-1]); kpl:=format('%14.5f',[Z[i]]);
mem1APIN2.Lines.Add('A('+ikpl+')='+kpl); end; end; end.
```

Наукове видання

ЩЕРБАНЬ В.Ю., КРАСНИТСЬКИЙ С.М.,
АСТІСТОВА Т.І., ЯХНО В.М.

МЕТОДИ ПРЕДСТАВЛЕННЯ, ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Підписано до друку 15.07.2023 р.
Формат 60x84/16. Папір офсетний.
Ум. друк. арк. 27,43.
Наклад 200 прим.

Видано ТОВ "Фастбінд Україна"
Свідоцтво про внесення суб'єкта видавничої справи до
державного реєстру видавців, виготівників
і розповсюджувачів видавничої продукції
ДК 6324 від 31.07.2018 р.