МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

І. В. Кравченко,
В. І. Микитенко,
Г. С. Тимчик

КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ: СИСТЕМИ І ПРОЦЕСИ

Підручник

Затверджено Вченою радою КПІ ім. Ігоря Сікорського
як підручник для здобувачів ступеня бакалавра
за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Рецензенти: *Заболотна Наталія Іванівна, доктор техн. наук, професор Вінницького технічного університету*

Ткачук Андрій Геннадійович, канд. техн. наук, зав. кафедри автоматизації та комп'ютерно-інтегрованих технологій Державного університету «Житомирська політехніка»

Відповідальний редактор *Чиж Ігор Генріхович, доктор техн. наук, проф.*

Гриф надано Вченої радою КПІ ім. Ігоря Сікорського (протокол №4 від 27.06.2022 р.)

Розглянуто теоретичні засади комп'ютерного моделювання, технології проведення обчислювальних експериментів, обчислювальні похибки, методики та особливості комп'ютерного моделювання в системах комп'ютерної математики (СКМ) «MATHCAD», «MATLAB» для моделювання процесів та приладів. Викладені теоретичні та практичні питання розв'язання алгебраїчних та нелінійних рівнянь, систем лінійних алгебраїчних та систем нелінійних рівнянь, проведення глобальної та ковзаючої інтерполяції, апроксимації, чисельного інтегрування та диференціювання, створення діалогової графічної оболонки для проведення комп'ютерних експеріментів в СКМ «MATLAB». Може бути корисний студентам та фахівцям технічних спеціальностей для набуття навичок застосування інформаційних технологій у вигляді сучасних СКМ в навчальній та науково-технічній практиці.

Реєстр. № П 21/22-025. Обсяг 10.7 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Провідні науковці сьогодні стверджують, що для ефективного розв'язання науково-технічних проблем сучасний інженер повинен [1; 2]:

- володіти фундаментальними інженерними знаннями, необхідними для розуміння принципів функціонування систем;
- володіти методами системного аналізу і керування проєктами;
- знати основні класи і принципи побудови моделей, методи моделювання систем, методи та етапи їх формалізації та алгоритмізації;
- вміти вибирати та використовувати методи математичного моделювання при проектуванні та експлуатації систем, розробляти схеми алгоритмів для дослідження та проектування технічних, технологічних, організаційних, інформаційних систем;
- знати і вміти застосовувати одну або декілька систем моделювання, проектування, мов програмування;
- мати уявлення про сучасний стан і перспективи розвитку інформаційних технологій, систем управління та систем обробки інформації;
- бути спроможним приймати рішення за результатами моделювання.

Дисципліна «Комп'ютерне моделювання процесів та систем» для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» викладається з лекційним обсягом 36 академічних годин. Місце дисципліни в навчальному плані не дозволяє використовувати елементи теорії систем, спектрального та імовірнісного аналізу, проектування систем автоматизації, інших спеціальних профільних фахових дисциплін, бо ці дисципліни ще не викладені студентам.

Комп'ютерному моделюванню присвячено значна кількість джерел. Більшість з них є академічними виданнями, які розглядають теоретичні питання моделювання [3; 4]. Серед вітчизняних видань україномовними навчальними є джерела, які присвячені імітаційному моделюванню [1], питанням моделювання механічних статичних та динамічних систем з орієнтацією на студентів старших курсів [5; 6]. Питання комп'ютерної математики викладені у вигляді фундаментальних дисциплін з обсягом 10-15 кредитів та орієнтацією на математичні спеціальності [7 – 9], ІТ-дисциплін з

описом можливостей систем комп’ютерної математики (СКМ) [10; 11] чи методичних розробок з практичним застосування СКМ для розв'язання математичних та технічних задач [12 – 14].

Метою підручника є поєднання в одному виданні базових теоретичних засад комп'ютерного моделювання, обчислювальних експериментів незалежно від того, які програмні засоби моделювання використовуються, та практичних засобів комп'ютерного моделювання, які реалізовані в сучасних СКМ.

В підручнику викладено відомості щодо принципів побудови моделей процесів та систем, методи та етапи їх формалізації й алгоритмізації, планування та проведення обчислювального експерименту, особливостей інженерних задач моделювання, базових чисельних методів комп'ютерного моделювання: розв’язання рівнянь, систем рівнянь, інтерполяції та апроксимації, чисельного інтегрування та диференціювання. Детально розглянуті можливості, особливості застосування технологій математичного моделювання засобами СКМ MathCAD, Matlab. Матеріал забезпечений практичними прикладами.

В розділі 1 розглядаються визначення та класифікація моделей, різновиди підходів до створення моделей, задачі моделювання, характеристики моделей, похибки моделей, питання проведення інженерного обчислювального експерименту, створення графічної оболонки для реалізації діалогу при проведенні комп’ютерного експерименту в СКМ Matlab.

В розділі 2 розглядаються основи теорії похибок комп'ютерних обчислень: джерела та типи похибок, правила запису наближених значень, округлення чисел, прямі та зворотні обчислення похибок, - загальний алгоритм чисельних методів, крок ітерації, критерії останову чисельних методів, збіжність чисельних методів

Чисельним методам розв’язання рівнянь гарантованої та умовної збіжності присвячено розділ 3.

В розділах 4, 5 розглянуто чисельні методи розв'язання систем лінійних алгебраїчних рівнянь та систем трансцендентних рівнянь, відповідно.

Розділ 6 присвячено наближенню функцій методами інтерполяції та апроксимації. Розглядаються теоретичні засади методики розв'язання задач глобальної поліноміальної, ковзаючої сплайн інтерполяції, апроксимації.

Розділи 7 розглядає методи квадратур Ньютона-Котеса, Гауса та їхньої точності для обчислення визначених інтегралів.

Чисельному диференціюванню аналітичних та сітчастих функції присвячено розділ 8. Розглянуто похибки розділених різниць, вибір оптимального кроку диференціювання, реалізації чисельного диференціювання на основі глобальної інтерполяції за Лагранжем, Ньютоном та основі формул «ковзаючої» інтерполяції.

Розділи 3 – 8 містять матеріал щодо засобів, можливостей, особливостей застосування, методиці розв'язання відповідних завдань в СКМ MathCAD та Matlab.

Підрозділи 1.1 - 1.3 підручника базуються на матеріалах лекцій [24] кредитного модуля з «Моделювання інформаційно-вимірювальних систем» , підрозділи 1.4, 3.3, 4.1, 5.1, 6.4, 6.7, 7.6 – на матеріалах практикуму [12] кредитного модуля «Комп'ютерне моделювання оптико-електронних приладів», які викладались в НТУУ «КПІ ім. Ігоря Сікорського» авторами підручника.

Приклади в підручнику викладені в редакції для версії MathCAD 15 та Matlab 9X. Види вікон діалогу та синтаксис команд в інших версіях пакетів можуть дещо відрізнятись від наведених у даному підручнику.

Видання призначено для допомоги студентам в самостійному вивченні навчальної дисципліни, в інформаційному та методичному забезпеченні курсу, в набутті навичок застосування інформаційних технологій у вигляді сучасної СКМ в навчальній та науково-технічній практиці. Підручник може використовуватись для самостійної роботи та дистанційного навчання.

1. ЗАСАДИ ТЕОРІЇ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

1.1. Визначення та класифікація

Зміст понять **модель, моделювання** в різних сферах науки та техніки не є тотожними. Але незважаючи на це, можна виокремити одну визначальну спільну властивість: **модель** завжди в тій чи іншій мірі імітує або замінює оригінал.

В філософському розумінні **моделювання** є однією з категорій теорії пізнання, одним з методів наукових досліджень в багатьох сферах людської діяльності.

Стадії пізнання, на яких відбувається така заміна, а також форми відповідності моделі і оригіналу можуть бути різними. **Моделювання** можна розглядати як пізнавальний процес, що містить переробку інформації, яка надходить із зовнішнього середовища, про те, що відбуваються в цих явищах. В результаті у свідомості з'являються образи, відповідні об'єктам. **Моделювання** можна розглядати як побудову деякої системи-моделі, пов'язаної певними співвідношеннями подоби з системою-оригіналом. В цьому випадку відображення однієї системи в іншу є засобом виявлення залежностей між двома системами, відображеними у співвідношеннях подоби, а не результатом безпосереднього вивчення [1; 15].

В процесі моделювання модель виступає одночасно і як засіб, і як об'єкт досліджень, який знаходиться у певному відношенні **подібності** до модельованого об'єкту.

Подібність – це взаємно-однозначна відповідність між двома об'єктами, коли відомі функції переходу від параметрів одного об'єкту до параметрів іншого, а їхні математичні описи можуть бути зроблені тотожними [15].

В основі моделювання систем знаходиться **теорія моделювання** – теорія заміщення одних об'єктів (оригіналів) іншими об'єктами (моделями).

Сучасний термін **модель** походить від латинських слів **modulus** – образ, **modelium** – міра.

Моделювання – це спосіб дослідження будь-яких явищ, процесів або об'єктів шляхом побудови й аналізу їх моделей.

Моделювання можна визначити як метод опосередкованого пізнання, при якому досліджуваний об'єкт-оригінал знаходиться в деякій відповідності з іншим об'єктом-моделлю, причому модель здатна в тому чи іншому відношенні заміщати оригінал на деяких стадіях пізнавального процесу [1; 15].

Моделювання являє собою процес заміни однієї системи (об'єкта-оригіналу або процесу) іншою системою з метою отримання інформації про цей об'єкт-оригінал шляхом проведення експериментів з отриманою моделлю.

В англомовних джерелах процес **моделювання** визначається двома термінами: побудова моделі оригіналу (**modeling**) та дослідження параметрів оригіналу з використанням побудованої моделі (**simulation**).

Будь-яка **модель** відображає лише деякі сторони оригіналу (параметри, які цікавлять розробника оригіналу), завдяки чому модель набуває певної ідеалізованої форми. Тому, досить часто для всебічного вивчення оригіналу доводиться будувати і досліджувати цілу низку моделей.

Складність **моделювання** як процесу полягає у відповідному виборі такої сукупності моделей, які заміняють реальний об’єкт у потрібному відношенні.

В навчальних та наукових джерелах [1; 15] модель визначають як:

- сутність / об'єкт, який відображає процеси, що протікають в реальних системах за допомогою математичних або натурних засобів;
- існуючу реально, або розроблену теоретично систему, яка заміщує об’єкт-оригінал, таким чином, що дає уявлення про властивості об’єкта.

Юридично в Україні рекомендовані терміни, які визначені в Державних стандартах України [1; 6; 7].

ДСТУ 2226-93 визначає моделювання як «подання різних характеристик поведінки фізичної чи абстрактної системи за допомогою іншої системи».

ДСТУ 2938-94 визначає моделювання (симуляцію) як «використання системи обробки даних для представлення спеціально обраних поведінкових характеристик фізичної або абстрактної системи».

В моделях відображаються процеси у вигляді взаємозв'язків показників процесу і на цій основі здійснюються оцінки характеристик (залежностей) та / або параметрів процесів досліджуваних систем.

Модель є умовним відображенням реальної системи. Умовність пов'язана з можливістю формалізації опису процесу і взаємозв'язків його показників.

При аналізі складних систем розрізняють **системний** та **індуктивний** підходи [3].

Індуктивний підхід розглядає систему шляхом переходу від частини до загального, синтезує систему шляхом злиття її компонент, які розроблюються окремо.

Системний підхід застосовує послідовний перехід від загального до частого, виділяючи характеристики, які описують досліджувані властивості об’єкта.

Процес синтезу моделі М на основі класичного (**індуктивного**) підходу наведено на рис. 1.1 а. Реальний об'єкт, розбивається на окремі підсистеми. Для моделювання вибираються вхідні дані Д та ставляться цілі моделювання Ц, які відображають окремі боки процесу моделювання. Виходячи з цієї цілі формується компонента К моделі. Сукупність компонент об’єднується в модель М. Розробка моделі М на базі класичного підходу означає сумування окремих компонент в єдину модель, причому кожна з компонент вирішує свої власні задачі та ізольована від інших частин моделі.

Системний підхід означає, що кожна система S є інтегрованим цілим навіть тоді, коли складається з окремих підсистем (рис. 1.1 б). На базі вхідних даних Д, обмежень ззовні або з обмежень можливостей реалізації моделі, формулюються вимоги Т до моделі системи S. На базі цих вимог формуються підсистеми П, елементи Е та проводиться вибір В складових системи за критеріями КВ.

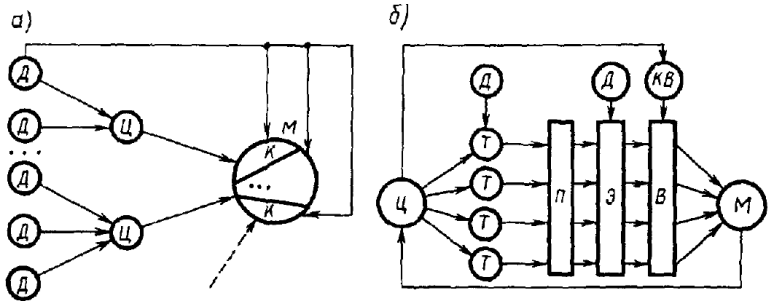


Рисунок 1.1 – Види моделювання: а – індуктивне, б – системне [3]

Класифікація моделей може здійснюватися за різними критеріями і носить умовний характер.

Наведемо класифікацію та характеристики моделей, які запропоновані проф. Трусовим В. П., проф. Струтинським В. Б., проф. Томашевським В. М., проф. Совєтовим Б. Я. [1; 3; 5; 15].

Моделі поділяються на два великих класи: **предметні** (матеріальні, експериментальні) та **абстрактні** (ідеальні, теоретичні).

Предметні моделі утворюються з сукупності матеріальних об'єктів і представляють собою реально існуючі пристрої двох основних типів.

Предметні моделі **першого типу** (**фізичні**) відтворюють оригінал у спрощеному вигляді, причому природа матеріальних елементів (складових частин) цих моделей може відрізнятися від природи елементів з яких складається об'єкт моделювання. Прикладом предметної фізичної моделі є макет. **Фізичні моделі** створюються на основі **теорії подібності**, причому подібність здійснюється за тими параметрами, які є суттєвими для дослідника. Так, наприклад, для вивчення опору руху корабля потрібна модель, зовнішні форми якої подібні до зовнішніх форм оригіналу, для дослідження міцності цього ж корабля – модель, що імітує його силовий каркас.

Макетні моделі – це реально існуючі моделі, що відтворюють модельовану систему в певному масштабі. Іноді такі моделі називаються **масштабними**. Параметри моделі і системи відрізняються між собою. Числове значення цієї відмінності називається **масштабом моделювання**, або **коефіцієнтом схожості**. Ці моделі розглядаються в рамках теорії подібності, яка в окремих випадках передбачає геометричну схожість оригіналу і моделі для відповідних масштабів параметрів. Прості макетні моделі - це пропорційно зменшені копії існуючих систем, які відтворюють основні властивості системи або об'єкта залежно від мети моделювання. Макетні моделі широко використовуються під час вивчення фізичних та аеродинамічних процесів, гідротехнічних споруд і багатьох інших технічних систем.

Предметні моделі **другого** типу – моделі на основі **методу аналогій** – ґрунтуються на тому факті, що різні фізичні явища описуються за допомогою однакового математичного апарату. Наприклад, коливальні процеси в механічних та електричних системах описуються однаковими звичайними диференціальними рівняннями другого порядку. В основу аналогового

моделювання покладено збіг математичних описів різних об'єктів. Прикладами аналогових моделей можуть служити електричні і механічні коливання, які з точки зору математики описуються однаковими співвідношеннями, але є якісно відмінним фізичними процесами. Прі деяких припущеннях, аналогічними можна вважати процеси розповсюдження тепла в тілі, дифузії домішок і просочування рідини. Це означає, що замість проведення досить складного і дорогого експерименту з механічною системою можна провести простіший експеримент з електричною системою, яка в цьому випадку і буде виступати моделлю.

Перевагами предметних моделей є висока адекватність моделі реальній системі та висока точність результатів.

Недоліками предметних моделей є висока вартість створення моделі; великі часові витрати; необхідність доопрацювання окремих вузлів реальної системи для проведення натурних експериментів. Наприклад, дослідження характеристик надійності автомобілів (crash test).

Дослідження предметних моделей проводять методами **натурного** та **фізичного** моделювання.

Натурне моделювання включає проведення експерименту з реальним об’єктом. Вони ґрунтуються на вимірюванні характеристик процесів, що відбуваються в реальних системах. Відмінність моделювання від реального процесу полягає в тому, що в моделюванні можуть досліджуватися критичні ситуації та межі стійкості процесу.

Іншим видом предметного моделювання є **фізичне**, яке відрізняється тим, що експеримент проводиться на установках, які зберігають природу явищ та мають фізичну подобу відносно досліджуваних параметрів. Наприклад, дослідження властивостей судна на його макеті.

Абстрактні (ідеальні) моделі описують об’єкт дослідження за допомогою певної мови. Абстрактність цих моделей проявляється в тому, що компонентами моделі є логічні поняття (образи, кресленики, схеми, графіки, рівняння, алгоритми), а не фізичні елементи. Ці моделі мають форму залежностей між групами параметрів досліджуваного об’єкта. Серед абстрактних моделей розрізняють **гносеологічні**, **інформаційні**, **сенсуальні** моделі.

Гносеологічні моделі призначені для дослідження об’єктивних законів природи (наприклад, моделі сонячної системи, біосфери, світового океану

тощо). **Інформаційні** моделі описують поведінку оригіналу, а не імітують його. **Сенсуальні** моделі – це моделі, що впливають на почуття людини (музика, мистецтво тощо).

При спостереженні за об'єктом-оригіналом в голові дослідника формується якийсь уявний образ об'єкта, його ідеальна модель, яку називають **когнітивною** (уявною, такою, що сприяє пізнанню).

Когнітивні моделі суб'єктивні, бо формуються «в голові» дослідника на основі його знань і досвіду. Отримати уявлення про когнітивну модель можна тільки описавши її в знаковій формі. **Когнітивна** модель, яка описана природною мовою називається **змістовною** моделлю. Не можна стверджувати, що **когнітивні** та **змістовні** моделі еквівалентні, бо перші можуть містити елементи, які дослідник не зможе або не хоче сформулювати.

Концептуальною моделлю називають **змістовну** модель, при формулюванні якої використовуються поняття і терміни предметних областей.

Концептуальні моделі мають за мету виявлення причинно-наслідкових зв’язків, які характерні для об’єкта дослідження і які є суттєвими в рамках дослідження. Один і той же об’єкт може бути представлений різними концептуальними моделями, наприклад, одна концептуальна модель може відображати часові аспекти функціонування системи, а інша – вплив відмов на працездатність системи.

ДСТУ 2832:2017 [9] дає визначення **концептуальної** моделі у вигляді «подання характеристик предметної сфери за допомогою об’єктів та відношень між об’єктами», яке не збігається з усталеною науковою термінологією.

Виділяють три види **концептуальних** моделей: **логіко-семантичні**, **структурно-функціональні** та **причинно-наслідкові**.

Логіко-семіотична модель є описом об'єкта в термінах відповідних предметних областей знань, що включає всі відомі логічно несуперечливі затвердження і факти.

В **структурно-функціональній** моделі об'єкт розглядається як цілісна система, яку розчленують на окремі елементи або підсистеми. Частини системи описують зв'язки структурних елементів, їхню підпорядкованість, логічну послідовність. Для представлення подібних моделей застосовують схеми, карти та діаграми.

Причинно-наслідкова модель використовується для пояснення та прогнозування поведінки об'єкта. Дані моделі орієнтовані в основному на виявлення головних взаємозв’язків між складовими елементами досліджуваного об'єкта, визначення того, як зміна одних факторів впливає на стан компонентів моделі, розуміння, як в цілому буде функціонувати модель і чи буде вона адекватно описувати об’єкт.

Формальна модель є представленням **концептуальної** моделі за допомогою однієї або декількох формальних мов (наприклад, мов математичних теорій або алгоритмічних мов).

Найбільш значимою **формальною** моделлю є **математична модель**.

Математичне моделювання являє собою процес заміни системи математичною моделлю і реалізації цієї моделі на основі розробленого алгоритму. Основне призначення математичних моделей полягає у формуванні структури (синтезі) систем і у відтворенні процесів функціонування останніх для оцінки їх характеристик, властивостей.

Під **математичною моделлю** розуміють будь-який математичний опис, що відображає з потрібною точністю структуру та/або процес функціонування деякої реальної системи в реальних умовах.

Саме через **математичну модель** ДСТУ 2938-94 визначає моделювання як «метод досліджень, який реалізується на ЕОМ та передбачає заміну реального процесу його математичною моделлю» [17].

Математична модель концентрує у вигляді математичних співвідношень сукупність знань, уявлень та гіпотез про об’єкт дослідження. Вона завжди описує поведінку реальної системи лише наближено, оскільки наші знання не є абсолютними, а гіпотези та припущення не враховують усі фактори. Такі моделі відносяться до класу **гомоморфних** моделей (**макромодель**), під якими розуміють моделі, що відображають лише основні властивості об’єкту дослідження. Між гомоморфною моделлю та оригіналом відсутнє повне, поелементне відображення.

В **ізоморфних** моделях спостерігається повна поелементна відповідність між моделлю та оригіналом (**повна** модель). В **повних** моделях фігурують фазові змінні, що характеризують стани всіх наявних між елементних зв'язків (тобто стану всіх елементів проектованого об'єкта).

При побудові моделі дослідник завжди виходить з поставлених цілей, враховує тільки найбільш істотні для дослідження фактори. Тому будь-яка

модель не є тотожною об'єкту оригіналу і, отже, **неповна**. Інші фактори, незважаючи на свій відносно малий вплив на поведінку об'єкта в порівнянні з вибраними факторами, в сукупності все ж можуть призводити до значних відмінностей між об'єктом і його моделлю. **Повна** модель, буде повністю тотожна оригіналу. Цю думку висловили Артуро Розенблют і Норберт Вінер: «Найкращою моделлю кота є інший кіт, а ще краще - той же самий кіт».

За характером відтворюваних сторін об'єкта проектування виділяють:

- **субстанціональні** моделі, які характеризують простір можливих станів об'єкта, приклади: довідники, описи типових проектних рішень, технологічних операцій;
- **функціональні** моделі, які характеризують об'єкт тільки в аспекті певних його відносин з середовищем або іншими об'єктами,
- **структурні** моделі, які характеризують внутрішню організацію об'єктів;
- **змішані** моделі.

Структурні моделі поділяються на **топологічні** й **геометричні**.

Топологічні моделі відображають склад і взаємозв’язки між елементами об'єкта. Найчастіше їх застосовують для опису об'єктів, які складаються з великого числа елементів, при розв’язанні задач прив'язки конструктивних елементів до певних просторових позицій (наприклад, задачі компонування обладнання, розміщення деталей, трасування з'єднань) чи до відносних моментів часу (наприклад, при розробці розкладів руху громадського транспорту, технологічних процесів). Топологічні моделі можуть мати форму графів, таблиць (матриць), списків тощо.

Геометричні моделі відображають геометричні властивості об'єктів, у них додатково до інформації про взаємне розташування елементів містяться дані про форму деталей. Геометричні моделі можуть виражатися сукупністю рівнянь ліній і поверхонь; алгебричних співвідношень, які описують області, що складають тіло об'єкта; графами і списками, що відображають конструкцію з типових конструктивних елементів, тощо. Геометричні моделі застосовують при розв’язанні задач конструювання в машинобудуванні, приладобудуванні, радіоелектроніці, для оформлення конструкторської документації, при задані вихідних даних на розробку технологічних процесів виготовлення деталей.

Функціональні моделі пов'язані з функціональним аспектом проектування систем і відображають закономірності процесів їх

функціонування. Ці моделі видаються як систем рівнянь, що описують відповідні процеси: електричні, механічні, теплові, процеси перетворення інформації тощо.

Залежно від складності модельованої системи в ієрархії функціональних моделей виділяють три рівня моделей – **мікрорівень, макрорівень (схемний)** і метарівень (**системний**).

На **мікрорівні** використовують математичні моделі, які описують фізичний стан і процеси в суцільних середовищах. Для побудови таких моделей застосовують апарат рівнянь математичної фізики. Наприклад, диференціальні рівняння в часткових похідних (рівняння електродинаміки, теплопровідності, пружності, газової динаміки тощо). В якості залежних змінних на мікрорівні можуть використовуватися фазові змінні, такі як напруженості полів, електричні потенціали, тиску, температури, концентрації частинок, щільності струмів, механічні напруги і деформації.

На **макрорівні** математичні моделі описують процеси в окремих елементах: деталях. В якості фазових змінних використовуються електричні напруги, струми, сили, швидкості, температури, витрати тощо. Функціональні моделі на макрорівні являють собою системи алгебраїчних або диференційних рівнянь.

На **метарівні** математичні моделі описують інформаційні процеси, що протікають в системах. Для побудови таких моделей використовують теорію автоматичного управління, теорію систем, математичну логіку, теорію кінцевих автоматів, теорію масового обслуговування.

За місцем використання в проектному процесі розрізняють **факторні** та **базові** макромоделі. **Факторні** моделі призначені для опису елементів на більш високому рівні. **Базові** моделі призначені для використання на тому ж рівні, на якому вони розробляються. Ці моделі слугують для скорочення розмірності задач даного рівня заміною фрагментів повної моделі.

Для опису технічних систем використовуються **аналітичні** та **імітаційні** математичні моделі.

Аналітичні моделі призначені для отримання функціональних залежностей шляхом послідовного застосування математичних формул та правил. В них модель представлена сукупністю аналітичних виразів, які відображають явні функціональні залежності між параметрами реальної

системи в процесі її роботи: лінійні і нелінійні рівняння, диференціальні та інтегральні рівняння, імовірнісні залежності.

При використанні **аналітичних** методів часто виникають труднощі пов’язані з неможливістю отримання розв’язку в аналітичній формі, що значно обмежує сферу їх застосування. В такому випадку проводиться дослідження аналітичних моделей в чисельному вигляді. Дослідження математичних моделей за допомогою чисельних методів полягає в заміні "неперервних" математичних операцій та відношень на відповідні дискретні аналоги: інтегралів на суми, похідних на їх аналоги у формі різницевих співвідношень, нескінченні суми на скінченні тощо.

Результати отримані в аналітичній формі є універсальними і мають велику цінність, оскільки вони дають змогу перевірити точність інших підходів.

Аналітичні методи при комп’ютерному моделюванні завжди використовують елементи чисельних методів.

Достоїнствами аналітичних моделей є простота і низька вартість моделі, можливість швидко отримати кількісні результати.

Недоліками аналітичних моделей є велике число припущень і обмежень, невисока точність результатів, відповідність результатів певним умовам, велика складність аналітичного опису функціональних залежностей.

Діючий в Україні стандарт ДСТУ ISO/IEC 2382:2017 [18] дає дуже широке визначення імітаційного моделювання як «використання системи опрацювання даних для подання вибраних поведінкових характеристик фізичної чи абстрактної системи. Приклад: подання повітряних потоків навколо деталей з аеродинамічним профілем за різних швидкостей, температур і тиску повітря».

Імітаційне моделювання – це сукупність методів алгоритмізації функціонування об'єктів досліджень, програмної реалізації алгоритмічних описів, організації, планування та виконання на ЕОМ обчислювальних експериментів з математичними моделями.

Імітаційні методи використовують, коли моделі являють собою опис об’єктів дослідження у формі алгоритмів. Вони адекватно відображають як структуру систем, що досягається ототожнюванням елементів системи з відповідними елементами алгоритмів, так і процеси функціонування системи, зображені в логіко-математичній формі. В імітаційних моделях часто

знаходять відображення багато деталей структури та функцій складних систем, які вимушено втрачаються або нехтуються в математично суворих моделях.

Достоїнствами імітаційного моделювання є висока адекватність між фізичною суттю описуваного процесу і його моделлю, можливість описати складну систему на досить високому рівні деталізації, значно більше областей дослідження, ніж аналітичне моделювання, відсутність обмежень відображення в моделі залежностей між параметрами моделі, можливість оцінки функціонування системи не тільки в стаціонарних станах, але і в перехідних режимах (процесах), одержання значної кількості даних про досліджуваний об'єкт (закон розподілу випадкових величин, числові значення абсолютні та відносні, і багато іншого).

Недоліками імітаційного моделювання є вища вартість і довший час розробки моделі в порівнянні з аналітичою, складніст оцінювання ступеню точності моделі, її адекватності досліджуваному процесу, відносно високі вимоги до кваліфікації розобника для написання програмної реалзації моделі.

Частина фахівців вважає [4; 19], що **імітаційне** моделювання найчастіше проводиться шляхом **імовірнісного**, **статистичного** моделювання, яке полягає в отриманні за допомогою комп'ютера статистичних даних про процеси, які відбуваються в модельованій системі, з наступною обробкою їх методами математичної статистики. Частина фахівців [1] вважає **імітаційні** моделі суто **алгоритмічними**.

В **комбінованих** моделях застосовується комбінація методів моделювання.

Моделі поділяють на **статичні** і **динамічні**. **Статична** модель описує явище або ситуацію в припущенні їх завершеності, незмінності (тобто в статиці). **Динамічна** модель описує як протікає явище або змінюється ситуація від одного стану до іншого (тобто в динаміці). При використанні динамічних моделей, як правило, задають початковий стан системи, а потім досліджують зміна цього стану в часі.

В залежності від специфіки зв'язку між характеристиками стану та вхідними даними розрізняють **детерміновані** та **стохастичні** (**імовірнісні**) математичні моделі. В **детермінованих** моделях у кожний фіксований момент часу характеристики стану системи однозначно визначаються через вхідні дані. Детерміновані моделі будують на основі фундаментальних теоретичних законів, таких як закони збереження енергії, маси, закони

термодинаміки, кінетики тощо. Усі величини, які входять до складу детермінованих моделей задають у вигляді конкретних чисел, векторів та функцій.

Якщо хоча б один параметр системи (об’єкта моделювання чи проектування) приймає випадкові значення, то таку побудовану математичну модель називають **стохастичною** або **імовірнісною**. У цьому випадку під однозначністю визначення характеристик стану системи розуміють однозначність визначення імовірнісного розподілу цих характеристик за заданими розподілами ймовірностей вхідних даних. Будь-якій реальній системі притаманні в тій чи іншій мірі випадкові флуктуації.

Модель називається **лінійною**, якщо для оператора моделі виконується принцип суперпозиції. У протилежному випадку модель називається **нелінійною**. Нелінійні моделі, в свою чергу, можна поділити на алгебричні та трансцендентні. У рівняннях нелінійної алгебраїчної моделі над змінними проводяться лише звичайні арифметичні операції та операції піднесення до степеня з раціональним показником, а в рівняння нелінійної трансцендентної моделі можуть входити інші функції над змінними (тригонометричні, логарифмічні тощо). Більшість реальних систем є нелінійними.

В **нестаціонарних** моделях вихідні параметри (розв’язання задачі) змінюються з часом, в іншому випадку– для **стаціонарних** моделей вихідні параметрі не залежать від змінної часу.

Неперервні моделі дають змогу визначити зміну вихідних параметрів в будь-який момент часу (для прикладу, модель, яка дає змогу визначити вихідні параметри аналогового електричний сигналу). **Дискретні** моделі оперують дискретною інформацією, тобто даними, отриманими в певні відліки часу (наприклад, моделі, що оперують з цифровою формою представлення електричного сигналу).

За способом використання експериментальних даних моделі поділяються на **апріорні** (розроблені теоретично) і **апостеріорні** (отримані в результаті обробки експериментальних даних).

За підходом до опису характерних властивостей об'єкта моделювання розрізняють моделі систем в **розподілених параметрах** і моделі систем в **зосереджених параметрах**.

За типом (напрямом) вирішуваних завдань моделі поділяються на **прямі**, **зворотні**, **ідентифікаційні**.

Найбільш часто вирішують так звані **прямі** завдання, постановка яких виглядає наступним чином: за даним значенням вхідного даного **x** при фіксованих значеннях параметрів a потрібно знайти рішення **y**. Процес розв'язання прямої задачі можна розглядати як математичне моделювання причинно-наслідкового зв'язку, властивого явищу. Тоді вхідний дане **x** характеризує «причини» явища, які задаються і варіюються в процесі дослідження, а шукане рішення **y** – «наслідок». Для того щоб математичний опис можна було застосовувати не до одиничного явища, а до широкого кола близьких за природою явищ, будують не одиничну математичну модель, а деякий параметричне сімейство моделей. Вибір конкретної моделі з цього сімейства здійснюється фіксацією значень параметрів моделі **a**. Наприклад, в ролі таких параметрів можуть виступати деякі з коефіцієнтів, що входять в рівняння. За допомогою вибору параметрів може проводитися вибір типу функціональної залежності між величинами. Якщо використовувані математичні моделі розбиті на класи, то параметром може служити і клас використовуваної моделі.

Велику роль відіграють **обернені** задачі, які складаються з визначення вхідного даного **x** за даним значенням **y** при обраних параметрах моделі **a**. Рішення зворотної задачі – це спроба з'ясувати, які «причини» **x** привели до відомого «наслідку» **y**. Як правило, зворотні завдання виявляються складніше для вирішення, ніж прямі.

Крім двох розглянутих типів завдань слід згадати ще один тип – задачу **ідентифікації**. У широкому сенсі завдання **ідентифікації** моделі – це завдання вибору серед безлічі всіляких моделей тієї, яка найкращим чином описує досліджуване явище. У такій постановці ця задача виглядає як практично нерозв'язна проблема. Найчастіше задачу ідентифікації розуміють у вузькому сенсі, як завдання вибору із заданого параметричного сімейства моделей конкретної математичної моделі (за допомогою вибору її параметрів a), з тим щоб оптимальним в сенсі деякого критерію чином узгодити слідства з моделі з результатами спостережень.

1.2. Характеристики моделей

Під **адекватністю** як правило розуміють правильний кількісний опис об’єкта за вибраними характеристиками з необхідною точністю, відображення заданих властивостей моделі із заданою точністю, коли

результати моделювання можуть бути використані для прогнозування поведінки або властивостей досліджуваного об'єкта чи процесу.

При цьому **адекватність** моделі залежить від цілей моделювання та прийнятих критеріїв. Враховуючи закладену при створенні неповноту моделі, можна стверджувати, що повністю адекватна модель принципово неможлива.

Слід підкреслити відносний характер поняття **адекватності**. Наприклад, якщо вивчається реакція об’єкта на зовнішні збурення того чи іншого класу, то модель, яка є адекватною відносно одного класу збурень, може виявитися неадекватною відносно іншого класу збурень.

Адекватність (відповідність) об’єкту-оригіналу з боку обраної системи його характеристик є найважливішою характеристикою моделі. Одним з критеріїв адекватності є характеристика **точності** моделі.

Точність моделі визначається як ступінь співпадіння вихідних параметрів моделі та об’єкта-оригіналу. Оберненою величиною до точності моделі є її **похибка**.

Похибка моделі не може бути меншою за найбільше значення похибки одного з вихідних параметрів математичної моделі.

Точність моделі залежить від умов функціонування об’єкта-оригіналу в просторі зміни вхідних параметрів. Область адекватності моделі (рис. 1.2) визначає область в просторі зміни вихідних параметрів, всередині якої наявна похибка моделі не перевищує наперед задане граничне значення похибки.

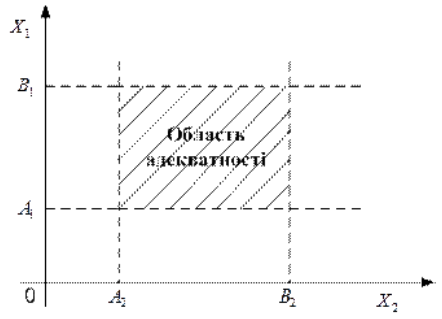


Рисунок 1.2 – Приклад області адекватності моделі [5]

Похибки, які виникають при математичному моделюванні, мають різний характер і величину. Для їх визначення та врахування здійснюється аналіз причин і місця виникнення помилок (рис. 1.3).

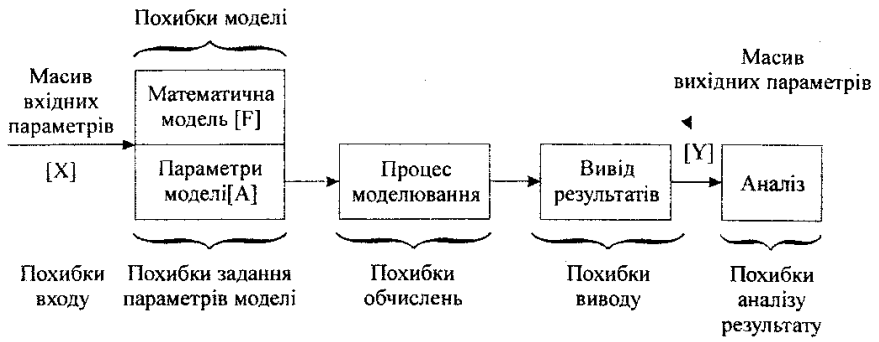


Рисунок 1.3 – Похибки моделювання [5]

Математична модель вважається точною, якщо абсолютні або відносні похибки не перевищують встановлених граничних значень.

Похибки моделі спричинені, як правило, неточностями, допущеними при виборі стратегії моделювання, а також особливостями алгоритму і обчислювальної процедури.

Похибки введення вхідних та проміжних параметрів поділяються на **грубі** (зі значенням граничної відносної похибки 10 % і більше), та на **несуттєві** похибки, Грубі похибки виникають від недостатньої інформації про дійсні вхідні параметри моделі, несуттєів похибки пов'язані із особливостями математичного аналізу вхідних параметрів.

Похибки методу пов’язані з тим, що будь-яка математична модель відтворює об’єкт оригінал наближено. Наприклад, використання нульових, дійсних променів теорії оптичних систем та диференційних хвильових рівнянь. Додаткові методичні похибки обумовлені застосуванням в математичній моделі чисельного методу. Наприклад, використання методу трапецій при чисельному інтегруванні вводить похибку обчислень, що відповідає кроку інтегрування в 4-му степені.

Похибки обчислень виникають при виконанні арифметичних операцій, зокрема, це є похибки обмеження та похибки округлення комп'ютерних обчислювальних операцій.

Комп'ютерні моделі є дискретними, результати моделювання представляються в матричному та векторному вигляді, тобто у вигляді масивів-таблиць. Методи обробки табличних значень призводять до похибок **виведення** значень вихідного параметра між двома елементами масиву (інтерполяція, апроксимація) або за межами масиву (екстраполяція).

Похибки виведення результатів моделювання можуть бути досить грубими і привести до помилкових висновків при аналізі результатів моделювання.

Похибки внаслідок високочастотних осциляцій значень результатів моделювання можуть суттєвим чином вплинути висновки моделювання. Необхідність їх врахування часто призводить до необхідності зміни стратегії моделювання, використання спеціальних алгоритмів і програмного забезпечення. Це має місце, наприклад, при чисельному інтегруванні «жорстких» диференціальних рівнянь.

Серед інших властивостей моделей слід відмітити:

- складність (або простота), яку оцінюють по загальній кількості елементів та зв'язків між ними;
- економічність, яка визначається затратами часу, матеріальних та фінансових коштів на розробку та використання моделі;
- цілісність, яка вказує на те, що модель відповідає цілісній системі;
- адаптивність, яка дозволяє пристосовувати модель до різних умов. Адаптивність визначається кількістю та складом врахованих в моделі зовнішніх і вихідних параметрів;
- універсальність характеризується областю застосування. Тобто, чим більша множина задач, які можна розв'язувати з використанням моделі, тим модель є більш універсальною.
- керованість моделі, яка витікає з необхідності забезпечити керування з боку експериментатора.;
- можливість розвитку моделі для передбачення розширення спектра досліджуваних функцій, кількості підсистем.

Вимоги простоти (економічності) моделі є протилежними до вимоги адекватності математичної моделі

1.3. Обчислювальний експеримент

Сучасний стан інформатики та комп'ютерної техніки суттєво підвищив роль комп'ютерного моделювання в науці та техніці. Сьогодні важко уявити собі проведення фундаментальних чи прикладних наукових досліджень без проведення математичного моделювання та комп'ютера. Для підкреслення важливості та ефективності такої технології навіть введений термін **обчислювальний** або **комп'ютерний експеримент.**

Обчислювальний або **комп'ютерний експеримент** – це дослідження властивостей об’єкта або явища шляхом розв’язування за допомогою комп’ютерних технологій задачі у вигляді математичної моделі цього явища чи об’єкта.

Проведення розрахунку моделі для різних наборів вхідних даних дає змогу дослідити роль та вплив різних факторів на протікання того чи іншого процесу або поведінку об’єкта, що дає змогу суттєво скоротити терміни, підвищити якість проектно-конструкторських робіт, виявити нові теоретичні та технічні якості досліджуваного процесу або об'єкта, знизити витрати на дослідження та проектування.

Значні практичні результати дозволяють говорити про обчислювальний експеримент як про нову технологію та нові методології наукових і прикладних досліджень.

Відзначимо деякі **достоїнства** обчислювального експерименту в порівнянні з натурним:

- обчислювальний експеримент, як правило, дешевше фізичного;
- в цей експеримент можна легко і безпечно втручатися, виокремлювати для досліджень окремий параметр чи групу параметрів та збурень, його можна повторити багаторазово і перервати в будь-який момент;
- у ході цього експерименту можна змоделювати умови, які не можна створити в лабораторії. В ряді випадків проведення натурного експерименту утруднене, а іноді й неможливо. Наприклад, при вивченні швидкоплинних процесів, важкодоступних або взагалі поки недоступних об'єктів (космічна техніка). Часто проведення повномасштабного натурного експерименту пов'язане зі згубними або непередбачуваними наслідками (надійність конструкцій споруд, суден) або з небезпекою для життя чи здоров'я людей. Нерідко потрібно дослідження і прогнозування результатів катастрофічних явищ (аварія ядерного реактора АЕС, пандемія);
- з допомогою цього експерименту можливо прогнозувати властивості нових, ще не створених конструкцій і матеріалів на стадії їх проектування.

Істотними **недоліками** обчислювального експерименту є те, що застосовність його результатів обмежена рамками прийнятої математичної

моделі, що неможливо провести обчислювальний експеримент в реальному часі.

Обчислювальний експеримент ніколи не зможе повністю замінити натурний, і майбутнє за їх розумним поєднанням. Само по собі комп'ютерне моделювання не дозволяє вирішити інженерну задачу, а лише дає в руки дослідника додатковий потужний інструмент. Використання комп'ютерної техніки не тільки не звільняє від необхідності глибоко осмислити проблему, а й змушує приділяти постановці завдання набагато більше уваги.

Особливості комп'ютерного розв'язання інженерної задачі

Поділ розв'язуваних із застосуванням комп'ютерів прикладних задач на інженерні, наукові, економічні тощо є до певної міри умовним. Роглянемо особливості саме інженерних задач, які виділено проф. Трусовим П.В. в [15].

Інженерні задачі мають яскраво виражену практичну спрямованість. Метою їх вирішення є створення нової конструкції, розробка нового технологічного процесу, мінімізація витрат на виробництво деякого виробу тощо. Тому для інженерних завдань характерна необхідність доведення результатів до конкретних чисел, графіків, таблиць, на підставі яких можна приймати рішення.

Інженерні завдання характеризуються значним обсягом виконуваної обчислювальної роботи, для них характерне використання досить складних математичних моделей і серйозного математичного апарату.

Як правило, інженерні завдання вирішують фахівці, що мають технічну освіту, але не є професіоналами в області розробки математичних методів і програмного забезпечення. Тому природно бажання цих фахівців використовувати готові обчислювальні методи і стандартне математичне програмне забезпечення.

Етапи проведення комп'ютерного експерименту

Розглянемо етапи проведення комп'ютерного експерименту, які виділені проф. Трусовим П. В. [15] та проф. Амосовим А. А. [2].

Розв'язання інженерної задачі та проведення комп'ютерного експерименту можна умовно поділити на наступні етапи:

- постановка проблеми;
- вибір або побудова математичної моделі;
- постановка обчислювальної задачі;

- передмашинний аналіз властивостей обчислювальної задачі;
- вибір або побудова чисельного методу;
- алгоритмізація та програмування;
- налагодження програми;
- перевірка адекватності моделі;
- проведення розрахунків, обробка та інтерпретація результатів;
- використання результатів та корекція математичної моделі.

Постановка проблеми. Спочатку прикладна задача буває сформульована в найзагальнішому вигляді: дослідити деякий явище, спроектувати пристрій із заданими властивостями, дати прогноз поведінки деякого об'єкта в певних умовах тощо. На етапі постановки обов'язковою є конкретизація первинного формулювання. Першорядна увага при цьому приділяється з'ясуванню мети дослідження. Від дослідника вимагається глибоке розуміння суті завдання і вміння сформулювати її так, щоб знайдене рішення було корисним і в той-же час могло бути отримано за допомогою існуючих методів і в реальні терміни. Невдала постановка проблеми може призвести до того, що тривалий і коштовний процес вирішення завдання завершиться отриманням даремних або тривіальних результатів. Цей дуже важливий і відповідальний етап завершується конкретним формулюванням проблеми мовою, прийнятою в даній предметній області.

Етап постановки та формулювання містить наступні роботи:

- обстеження чи вивчення об'єкта моделювання з метою виявлення основних факторів, механізмів, які обумовлюють його поведінку, визначення відповідних параметрів, що дозволяють описувати модельований об'єкт;
- збір та перевірка наявних експериментальних даних про об'єкти-аналоги, проведення, при необхідності, додаткових експериментів;
- аналітичний огляд літературних джерел, аналіз і порівняння між собою побудованих раніше моделей даного об'єкта;
- аналіз і узагальнення накопиченого матеріалу, розробляння плану створення моделі.

На основі зібраної інформації про об'єкт моделювання виконавці спільно з Замовником формулюють змістовну постановку завдання моделювання, яка, як правило, не буває остаточною і може уточнюватися в процесі розробки моделі. Всі наступні уточнення і зміни змістовної

постановки повинні носити непринциповий характер. Якщо об'єктом моделювання є технологічний процес, технічна система, то змістовну постановку задачі моделювання часто називають технічним завданням.

Технічне завдання є підсумковим документом, який завершує роботи обстеження.

Чим більш повну інформацію вдасться зібрати про об'єкт на етапі обстеження, тим більш чітко можна виконати змістовну постановку задачі, уникнути багатьох складностей на наступних етапах розробки моделі. Неконкретні і нечіткі вимоги до моделі можуть серйозно ускладнити процес здачі моделі замовнику. В цілому опрацювання технічного завдання може займати до 30% часу, відведеного на створення всієї моделі.

Враховуючи важливість розглянутого етапу, технічні завдання слід піддавати внутрішній і зовнішній експертизі незалежними експертами, які не беруть участі в його розробці.

На підставі змістовної моделі розробляється **концептуальна**, або «природничо-наукова» (фізична, хімічна, біологічна тощо) постановка завдання моделювання в термінах конкретних дисциплін.

Найбільші труднощі при формулюванні концептуальної постановки доводиться долати в моделях, що знаходяться на «стику» різних дисциплін. Відмінності традицій, понять і мов, що використовуються для опису одних і тих же об'єктів, є серйозними перешкодами, що виникають при створенні «міждисциплінарних» моделей.

На відміну від **змістовної концептуальна** постановка задачі моделювання, як правило, формулюється членами робочої групи без залучення представників замовника на основі розробленого на попередньому етапі технічного завдання з використанням наявних знань про об'єкт моделювання та вимог до майбутньої моделі.

При формуванні гіпотез про поведінку об'єкта, його взаємодії з навколишнім середовищем, зміні внутрішніх параметрів значною мірою проявляється мистецтво, досвід і знання, що накопичені членами робочої групи. Згідно з прийнятими гіпотезами визначається коло параметрів, що описують стан об'єкта, а також перелік законів, які керують зміною і взаємозв'язком цих параметрів між собою.

Вибір або побудова математичної моделі. Для подальшого аналізу досліджуваного явища або об'єкта необхідним є наявність формалізованого

опису об'єкта мовою математики, тобто математична модель. Часто існує можливість вибору моделі серед відомих і прийнятних для опису відповідних процесів, нерідко потрібна суттєва модифікація відомої моделі, а іноді виникає необхідність у побудові принципово нової моделі.

Розглянутий етап вибору моделі є чи не самим важливим і важким. Вдалий вибір математичної моделі є вирішальним кроком до досягнення мети. Одна з істотних труднощів такого вибору полягає в об'єктивному протиріччі між бажанням зробити опис явища як можна більш повним, що призводить до ускладнення моделі, і необхідністю мати просту модель, щоб була можливість її комп'ютерної реалізації.

Процес побудови математичних моделей трудомісткий та тривалий. Він пов'язаний з залученням знань з різноманітних областей в предметній царині, пов'язаної з об'єктом моделювання, так і в галузі математики, сучасних чисельних методів, програмування тощо. Відмінною особливістю математичних моделей є їх комплексність, пов'язана зі складністю модельованих об'єктів. Наприклад, при моделюванні процесів деформування конструкцій під дією прикладеного навантаження доводиться враховувати не тільки процеси, що відбуваються при деформуванні, а й процеси масопереноса, теплопереноса, а також пов'язані з ними зміни структури і властивостей матеріалу. У разі складних об'єктів задовольнити всім вимогам, що пред'являються в одній моделі зазвичай неможливо. Доводиться створювати цілий спектр моделей одного і того ж об'єкта, кожна з яких найбільш ефективно вирішує покладені на неї завдання.

В багатьох галузях знань можна виділити закони, справедливі для всіх об'єктів дослідження даної галузі знань, і співвідношення, що описують поведінку окремих об'єктів. До числа перших у фізиці і механіці належать, наприклад, рівняння балансу маси, кількості руху, енергії тощо. Вони справедливі за певних умов для будь-яких матеріальних тіл незалежно від їх будови, структури, стану, хімічного складу.

Співвідношення другого класу називають **визначальними**, або фізичними рівняннями, або рівняннями стану. Вони встановлюють особливості поведінки матеріальних об'єктів (наприклад, рідин, газів, пружних або пластичних середовищ тощо) при дії різних зовнішніх факторів. Наприклад, закон Гука в теорії пружності або рівняння Клапейрона для ідеальних газів. Співвідношення другого класу набагато менш вивчені, а в ряді випадків їх доводиться установлювати самому досліднику.

Саме похибки у виборі або встановлення визначальних співвідношень призводять до невірних результатів моделювання.

Побудова математичних моделей є суттєвою і дуже важлиою частиною природних і технічних наук. Це завдання, що вимагає від дослідника глибокого знання предметної області, високої математичної культури, досвіду побудови моделей, розвиненою інтуїції і багато чого іншого. Створення вдалою нової моделі стає завжди великим досягненням відповідної науки, а іноді й цілим етапом у її розвитку.

Постановка обчислювальної задачі. На основі прийнятої математичної моделі формулюють обчислювальну задачу або ряд задач.

Закінчена концептуальна постановка дозволяє сформулювати математичну постановку завдання моделювання. Математична постановка задачі моделювання – це сукупність математичних співвідношень, що описують поведінку і властивості об'єкта моделювання.

Сукупність математичних відносин визначає вид оператора моделі. Найбільш простим буде оператор моделі у випадку, якщо він представлений системою алгебраїчних рівнянь. Однак область застосування моделей подібного типу обмежена. Створення математичних моделей складних систем і процесів вимагає залучення великого обсягу знань, накопичених у предметній дисципліні та в суміжних областях. У більшості випадків оператор моделі включає в себе систему диференціальних або інтегро-диференціальних рівнянь. Для забезпечення коректності постановки завдання до системи рівнянь додаються початкові та/або граничні умови.

Необхідними на етапі постановки обчислювальної задачі є базові перевірки фізико – математичних співвідношень: розмірностей; порядків значень; напрямку зміни значень; граничних умов; математичної замкнутості.

Попередній аналіз властивостей обчислювальної задачі. На цьому етапі проводять попереднє (передмашинне) дослідження властивостей обчислювальної задачі. Особлива увага приділяється аналізу коректності формулювання обчислювальної задачі, тобто з'ясуванню питань існування та однозначності розв'язку, а також дослідженню стійкості розв'язку завдання до похибок вихідних даних. Таке дослідження, як правило, належить до компетенції професійних математиків. Проте інженерові корисно бути в курсі сучасного стану названих проблем, вміти самостійно проводити дослідження.

Особливу цінність мають аналітичні рішення – вони виявляються корисними не тільки для аналізу явища, але і як основа для тестових випробувань на етапі налагодження програми.

Вибір або побудова чисельного методу. Для комп'ютерного розв'язання обчислювальної задачі найчастіше використовуються чисельні методи. Часто рішення інженерної задачі зводиться до послідовного розв'язання стандартних обчислювальних задач, для яких вже розроблені ефективні чисельні методи. В таких випадках проводиться або вибір серед відомих методів, або адаптація відомих методів до особливостей розв'язуваної задачі. Однак, якщо обчислювальна задача є новою, то не виключено, що для її вирішення не існує готових методів. Побудова чисельного методу для такого завдання може виявитися дуже важкою проблемою, яка потребує фаху з обчислювальної математики. Для вирішення однієї і тієї ж обчислювальної задачі зазвичай може бути використано кілька методів. Необхідно знати особливості цих методів, критерії, за якими оцінюється їх якість, щоб вибрати метод, який дозволяє вирішити проблему найбільш ефективним чином.

Алгоритмізація та програмування. Як правило, вибраний на попередньому етапі чисельний метод містить тільки принципову схему вирішення завдання, яка не враховує багато деталей, без яких неможлива реалізація методу на ЕОМ. Необхідна докладна деталізація всіх етапів обчислень для того, щоб отримати робочий алгоритм для реалізації на ЕОМ. Складання програми зводиться до перекладу цього алгоритму на обраний мову програмування. Алгоритмізація та програмування дуже тісно пов'язані. Практика показує, що невеликі, на перший погляд, відмінності в програмах можуть призвести до значних відмінностей в їх ефективності.

Процес розробки надійного та ефективного програмного забезпечення є не менш складним, ніж попередні етапи. Успішне вирішення даного завдання можливе лише за впевненому володінні сучасними алгоритмічними мовами і технологіями програмування, знанні можливостей обчислювальної техніки, наявного програмного забезпечення, особливостей комп'ютерної реалізації методів обчислювальної математики, наявності досвіду розв’язання подібних завдань.

Налагодження програми. На цьому етапі виявляють і виправляють помилки в програмі.

Думка, що помилок у складеній ним програмі немає або ж вони можуть бути легко виявлені і виправлені, зазвичай, виявляється помилковою. Налагодження програми і доведення її до робочого стану часто виявляється більш тривалим і трудомістким процесом, ніж саме написання програми. Набуваючи певний досвід у складанні та налагодженні порівняно складних програм, користувач переконується в справедливості популярного афоризму: "У будь-якій програмі є принаймні одна помилка".

Після усунення помилок програмування необхідно провести ретельне тестування програми – перевірку правильності її роботи на спеціально відібраних тестових завданнях, що мають відомі рішення.

Перевірка адекватності моделі. Перевірка адекватності моделі переслідує дві цілі:

- переконатися в справедливості сукупності гіпотез, які сформульовані на етапах концептуальної та математичної постанови.
- встановити, що точність отриманих результатів відповідає точності, обумовленої в технічному завданні.

Перевірка розробленої математичної моделі виконується шляхом порівняння її результатів з наявними експериментальними даними про реальний об'єкт або з результатами інших достовірних моделей. У першому випадку говорять про перевірку шляхом порівняння з експериментом, у другому – про виконання тестової задачі.

Вирішення питання про точність моделювання залежить від вимог, що висуваються до моделі. У моделях, призначених для виконання оціночних розрахунків, задовільною вважається точність (10-15) %. У моделях, що використовуються в керуючих і контролюючих системах, необхідна точність може бути (1-2) % і навіть більше.

Розрізняють **якісне** і **кількісне** співпадіння результатів. Для якісно співпадіння потребується лише збіг деяких характерних особливостей в розподілі досліджуваних параметрів (наприклад, наявність екстремальних точок). Питання про кількісне порівняння можна ставити лише після задовільної відповіді на питання про якісну відповідність результатів. При кількісному порівнянні велике значення відіграє точності вихідних даних.

Дуже часто аналіз тестових результатів вказує на недосконаліть використаної математичної моделі і необхідність її корекції. В такому

випадку математичну модель модифікують (при цьому вона, як правило, ускладнюється) і починають новий цикл рішення задачі.

Неадекватність результатів моделювання можлива, по крайній мірі, з трьох причин:

1) значення параметрів моделі не відповідають допустимій області цих параметрів, обумовленою прийнятою системою гіпотез;

2) прийнята система гіпотез вірна, але константи і параметри у використаних визначальних співвідношеннях встановлені не точно;

3) невірна вихідна сукупність гіпотез.

Використання моделі. Обробка та інтерпретація результатів. Отримані в результаті розрахунків на ЕОМ вихідні дані, як правило, являють собою великі масиви чисел. Починаючий користувач часто намагається вивести ці масиви на друк з тим, щоб «потім провести їх аналіз». Зазвичай перший же досвід аналізу роздруківки, що містить сотні тисяч чисел, призводить до розуміння того, що ця робота непосильна для людини і слід постаратися покласти її на ЕОМ. Найчастіше першочерговий інтерес представляє лише невелика частина отриманої інформації (наприклад, значення однієї з функцій у виділених точках) або навіть деяка груба інтегральна характеристика (максимальне або мінімальне значення, оцінка енергії системи тощо). Для того щоб дослідник міг скористатися результатами розрахунків, їх необхідно представити у вигляді компактних таблиць, графіків або в іншій зручній для сприйняття формі.

Для правильної інтерпретації результатів розрахунків та оцінки їх достовірності від дослідника вимагається глибоке знання розв'язуваної інженерної задачі, ясне уявлення про використану математичну модель і розуміння особливостей застосовуваного обчислювального методу.

1.4. Діалог комп'ютерного експерименту

Найбільш ефективним та зручним є проведення комп'ютерного експерименту в інтерактивному середовищі (рис. 1.4), коли результати, кількісні параметри, графіки характеристик супроводжуються необхідною інформацією про умови досліджень, а керування ходом моделювання здійснюється зручними елементами керування (кнопками, списками, перемикачами, меню). СКМ Matlab має вбудовані засоби для створення подібного середовища.

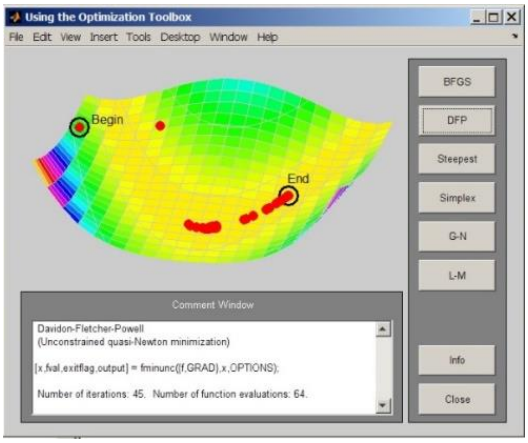


Рисунок 1.4 – Вигляд графічного вікна з елементами керування.

Розробляння інтерактивних застосунків використовує на об’єктну модель програмування Matlab та містить наступні основні етапи:

- створення діалогового вікна шляхом опису розташування потрібних елементів інтерфейсу у вікні;
- визначення послідовностей дій у вигляді функцій обробки, які будуть виконуватися при зверненні користувача відповідних об’єктів інтерфейсу, наприклад, при зміні стану перемикача.

Кінцевим результатом є готовий для запуску застосунок Matlab, який не потребує програмування при проведенні моделювання. Умови експерименту тощо змінюються елементами графічного інтерфейсу (**`GUI`**). Такі застосунки можуть складатися з кількох файлів. Запуск застосунків зазвичай проводиться написом імені основного модуля програми в командному рядку Matlab.

В Matlab розробляння **`GUI`** застосунків може проводитися «вручну» засобами вбудованої **низькорівневої дескрипторної** (**`handle`**) графіки або з використанням вбудованого середовища **`GUIDE`**, яке саме призначено для створення опису **`GUI`** шляхом візуального конструювання діалогових графічних вікон.

Низькорівнева дескрипторна графіка

Компоненти **`GUI`** графічних вікон базуються на об'єктній ієрархічній моделі програмування (рис. 1.5).

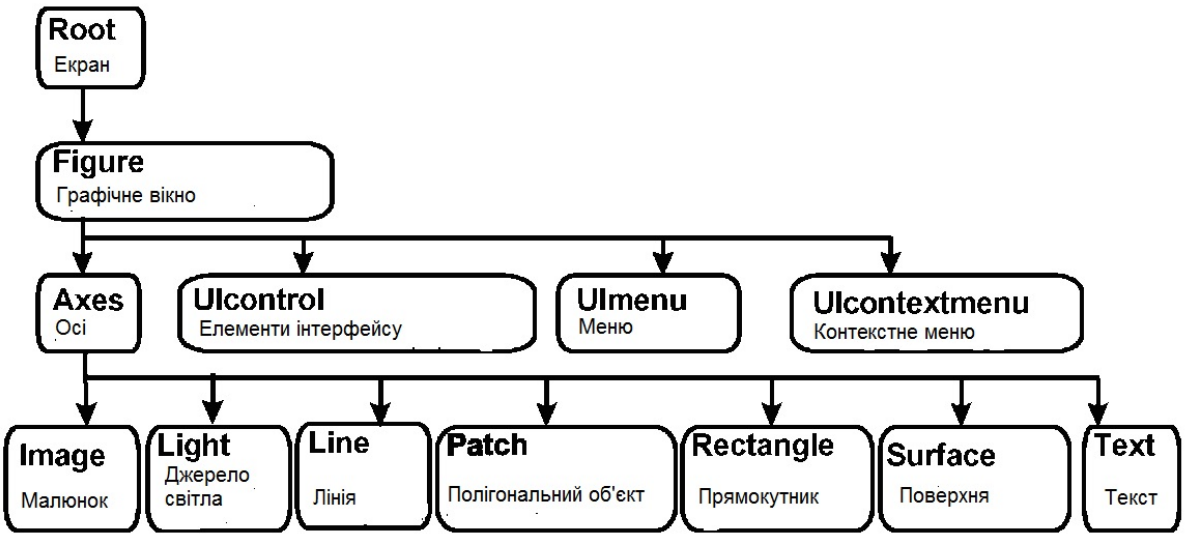


Рисунок 1.5 – Ієрархія графічних об’єктів Matlab [12]

Компоненти **GUI** є об’єктно-орієнтованими базовими елементами та визначаються своїми властивостями (**properties**).

Ієрархія об'єктів **GUI** виглядає наступним чином:

Root – вершина ієрархії. Об'єкт **Root** відповідає екрану комп’ютера.

Figure – вікна на екрані.

Uicontrol – універсальний елемент інтерфейсу користувача.

Text – статичний текст.

Edit – поле введення текстових даних користувача.

Checkbox – прапор.

Listbox – список.

Frame – групова панель, рамка.

Popupmenu – список, що розкривається.

Pushbutton – кнопка.

Togglebuttoon – перемикач.

Radiobutton – кнопка для вибору з кількох варіантів.

Slider – повзунок скролінгу.

Uipanel – панель для розташування елементів.

Uibuttongroup – група радіокнопок

Uimenu – меню інтерфейсу користувача, яке розташоване в верхній частині вікна **Figure**.

Uicontextmenu – меню інтерфейсу користувача, яке викликається правою кнопкою миші над елементом інтерфейсу.

Axes – область графіків у вікні **Figure**.

Image – зображення. Задаються у вигляді матриць.

Line – лінії.

Surface – поверхні. Задаються у вигляді матриць.

Light – джерело світла для об’єктів **Axes**.

Під час запуску застосунка Matlab автоматично присвоює кожному графічному об’єкту унікальний ідентифікатор **handle** (покажчик) для програмної ідентифікації об'єкта.

Ідентифікатор об’єкта **root** завжди дорівнює нулю. Ідентифікаторами об’єктів **figure** є цілі числа. Число-ідентифікатор за замовчанням відображається в заголовку вікна. Ідентифікаторами інших об’єктів є дійсні числа.

Об'єкт, який створений чи вказаний користувачем останнім називається поточним.

Для адресації поточних об’єктів існують наступні вбудовані покажчики:

gcf – покажчик на поточне вікно.

gca – покажчик на поточне поле графіка.

gco – покажчик на поточний об’єкт **GUI**.

gcbo – покажчик на графічний об’єкт, функція обробки якого виконується в даний час.

Об’єкт **Root** немає предків, а його потомками є графічні вікна – об’єкти **Figure**. Покажчик на **Root** завжди дорівнює нулю. Об’єкт **Root** автоматично створюється при запуску Matlab та слугує для визначення загальних властивостей. Властивість **units** об’єкта **Root** задає одиниці виміру розмірів компонентів **GUI.** За замовчанням розміри визначаються в пікселях екрану. Властивість **Pointer** об’єкта **Root** містить покажчик на графічне вікно, над яким знаходиться курсор або нуль, коли курсор знаходиться зовні графічних вікон.

Графічне вікно **Figure** є контейнером, в якому можуть розміщуватися його нащадки: осі (**Axes**), елементи інтерфейсу (**uicontrol**), меню вікна (**uimenu**), контекстні меню (**uicontextmenu**) тощо Властивості **Resize** об'єкта **Figure** дозволяє керувати можливістю змінювати розміри вікна. Значення **off** забороняє зміну розмірів вікна, значення **on** – дозволяє.

Вікна можуть бути двох типів: **звичайні (`normal`)** – вікна, між якими можна переключатися в будь-який момент, та **модальні** (діалогові, **`modal`**) – вікна, які не дозволяють перемикатися без закриття даного вікна. Властивість **`WindowStyle`** вікна **`Figure`** визначає тип вікна: **`normal`** за замовчанням чи **`modal`**.

Властивості **`currentAxes`** та **`Currentobj`** об'єкта **`Figure`** визначають покажчики на поточні осі та об'єкти, які розміщені в області вікна.

Властивість **`Currentcharacter`** об'єкта **`Figure`** містить символ, який користувач ввів з клавіатури.

Доступ до властивостей об'єктів **`Figure`** за замовчанням проводиться тільки з рівня предку **`Figure`**, тобто об'єкта **`Root`**.

Всі об'єкти для графічного зображення даних: **`Image`**, **`Light`**, **`Line`**, **`Patch`**, **`Rectangle`**, **`Surface`**, **`Text`**,- є потомками об'єкта **`Axes`**.

Властивість **`callbackobject`** містить покажчик на об'єкт, дія (властивість **`callback`**) якого обробляється на даний момент. Властивість **`CurrentFigure`** містить покажчик на поточне графічне вікно. Властивість **`PointerLocation`** містить положення курсору у вигляді вектора з двох елементів.

Створення об'єктів

Для створення об'єкту слід викликати функціє з його іменем. Наприклад, функція **`text`** створює об'єкт **`text`**, функція **`figure`** створює об'єкт **`figure`** тощо.

Більшість вбудованих команд Matlab побудови графіків автоматично створює необхідні об'єкти та надає доступ до визначення властивостей у вигляді пари аргументів «властивість, значення». Для того, щоб реалізувати вказану можливість слід не просто використати команду створення графіка, а записати команду у вигляді функції з присвоєнням значення змінній типу покажчик.

Наприклад, наступні рядки просто генерують відповідні зображення

```
plot(x,y); text(0.1, 0.1, 'приклад'),
```

наступні рядки генерують зображення та зберігають у відповідних змінних ідентифікатори об'єктів для можливої програмної зміни властивостей об'єктів:

```
hf=figure    %покажчик на вікно
ha=axes      %покажчик на поле графіка
hp=plot(x,y) %покажчик на лінію графіка
ht=text(0.1, 0.1, 'приклад')  % покажчик на текст.
```

Для видалення об’єкта слугує функція **delete**. Аргументом функції є покажчик на об’єкт, який планується знищити. Наприклад, **delete(gca), delete(ha)** видалить поточні осі **axes** разом всіма потомками.

Властивості об’єктів

Згідно з парадигмою об'єктного програмування властивості об’єктів визначають, в якому вигляді об’єкти виводяться на екран (табл. 1.1). Matlab надає два механізми для визначення властивостей. Властивості можна визначити під час створення об’єкту або змінити чи надати властивості вже існуючому об’єкту.

Таблиця 1.1. Властивості об’єкта осі [12]

Властивість	Опис	Значення
Box	Рамка навколо поля графіка	**on / off**
FontName	Назва шрифту	Рядок з назвою шрифту
FontAngle	Нахил шрифту осей	**normal / italic**
Fontsize	Розмір шрифту	Ціле число
FontWeight	Товщина шрифту	**normal,bold,light,demi**
GridLineStyle	Стиль ліній сітки	**-/ - -/ :/ -/ none**
LineWidth	Товщина ліній осей	Значення в пунктах (1/72 дюйма)
visible	Видимість осей	**on /off**

Таблиця 1.1. Продовження

Властивість	Опис	Значення
Color	Колір фона осей	Колір може бути задано чисельним значенням RGB, скороченою чи повною назвою: RGB Short Long **[1 1 0] y yellow** **[1 0 0] r red** **[0 1 0] g green** **[0 0 1] b blue** **[1 1 1] w white** **[0 0 0] k black**
Властивості осі X (Y)		
x(y)color	Колір осі	Див. **color**
X(Y)Dir	Напрям осі	**normal/ reverse**
X(Y)Grid	Перпендикулярність сітки	**on/off**
X(Y)AxisLocation	Розташування осі	**top / bottom (right / left** для осі Y**)**
X(Y)Lim	Межі змін	Вектор з двох компонентів, що є межами, [-1.5 2.3]
X(Y)Scale	Тип осі	**linear / log**
X(Y)Tick	Координати міток осі	Вектор з координатами розмітки, наприклад [0 13 5]
X(Y)TickLabel	Мітки осі	Вектор комірок з назвами розмітки (число комірок дорівнює довжині вектора з координатами розмітки), наприклад [**'zero'; 'one';'three'; 'five'**]

Для задання значення будь якої властивості графічного об’єкту призначена команда

Set(покажчик об’єкту, 'PropertyName',PropertyValue)

Аргументами функції є покажчик об’єкта та пара «властивість, значення».

Частина назв властивостей закінчується словом **Mode**, наприклад **YTickMode**. Такі властивості можуть мати два значення – **'auto'** (встановлюється за замовчанням) та **'manual'**. Режим **'auto'** забезпечує автоматичний підбір системою значення відповідної властивості. Наприклад, визначення конкретного значення вектора в **YTick** призводить до зміни значення **YTickMode** з **'auto'** на **'manual'**. Скинути властивості **YTick** можна установкою **YTickMode** в **'auto'**. Вищенаведене вірно для всіх властивостей, які закінчуються на ‘**Mode’**.

Наприклад, наступний код виводить графік залежності температури шляхом створення трьох об’єктів з визначенням деяких з властивостей поля графіка під час створення об'єкту (рис. 1.6):

```
days = ['Su';'Mo';'Tu';'We';'Th';'Fr';'Sa'   ];
temp = [21.1 22.2 19.4 23.3 23.5 21.1 20.0];
f= figure;
a=axes('Ylim',[16 26],'Xtick,1:7,'XtickLabel',days )
h=line( 1:7,   temp)
```

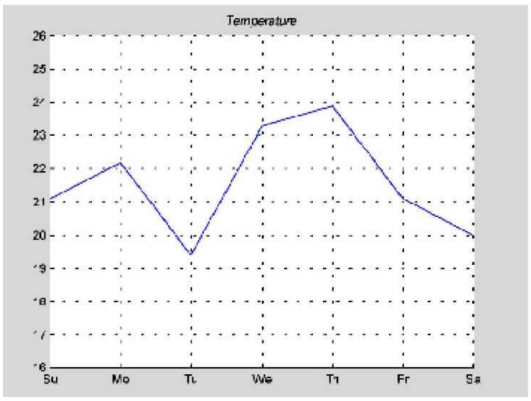


Рисунок 1.6 – Графік температурної залежності

В наведеному коді для даних використані змінні **days** – масив символів, який містить дні тижня, та **temp** – чисельний масив температур. Графічне вікно створюється функцією **figure** без аргументів, тобто зі значеннями за умовчанням. Для поля графіка **axes** під час створення визначаються заданий діапазон по осі **y** та задані мітки по осі **x**. Лінія

створюється як потомок об'єкту **axes** та має задані значення для даних. Для подальшого застосування покажчики об'єктів збережені в змінних **f, a, h**.

Прочитати значення властивостей об’єкта можна командою **Get(покажчик об’єкта <,'PropertyName'>).** Аргументами функції є покажчик об’єкта та назва властивості об'єкта.

За допомогою команд **uicontrol, uimenu, uicontextmenu** створюються об'єкти GUI, які керують введенням даних та процесом моделювання, та задаються їхні властивості (табл. 1.2). Значення властивостей, які не визначено безпосередньо в команді, призначаються за замовчанням.

Команда **<h>=uimenu('PropName',PropVal,...)** створює меню, задає його властивості та повертає покажчик **h** на нього.

Команда

<h>=uicontrol(’Parent’,<parent>,'PropName',PropVal,...)

створює елемент завданого типу, задає його властивості та повертає покажчик **h** на нього. Тип елемента визначається властивістю **Style**.

Команда **<h>=uicontextmenu('PropName',PropVal,...)** створює контекстне меню, задає його властивості та повертає покажчик **h** на нього. Контекстне меню викликається правою кнопкою миші та спливає безпосередньо над елементом.

Наприклад, команда

```
uicontrol('Style','edit','String','hello')
```

створює поле введення у поточному вікні.

Наступні рядки створюють два вікна та у першому створюють поле введення зі значенням у ньому 'hello' (рис. 1.7).

```
  fig1 = figure;
  fig2 = figure;
  uicontrol('Parent',fig1,'Style','edit',...
'String','hello');
```

Таблиця 1.2. Основні властивості елементів керування [12]

Властивість	Зміст	Значення
BackgroundColor	Колір фона	Вектор RGB чи службове слово: [**100**] / **'r' / 'red'**
BusyAction	Тип обробки переривання користувача при виконанні **callback** функції	**cancel / queue**
ButtonDownFcn	Ім'я **callback** функції, що виконується коли елемент відключений	Рядок
SliderStep	Величина зсуву повзунка	Вектор, два значення в діапазоні 0 ..1 [поле повзунка кнопка]
String	Назва елемента або пункти списків, що відображуються на екрані	Рядки багаторядкового тексту розділяються символами `\n '
Tag	Ім'я об'єкта	Рядок
Style	Тип об'єкта	pushbutton / togglebutton / radiobutton / checkbox / edit/ text / slider / frame / listbox / popupmenu
TooltipString	Спливаюча підказка	Рядок
UIContextMenu	Контекстне меню об'єкта	Покажчик
Units	Одиниці виміру розмірів об'єкта	pixels / normalized (0..1) /| inches / centimeters / points(1/72 inch) / characters
UserData	Дані користувача. (Визначення – **set**, показ – **get**)	Матриця
Visible	Видимість об'єкта	**on /off**
Value	Поточне значення об'єкта	Число чи вектор Check boxes – Max/ Min, List boxes – вектор номерів обраних пунктів, Sliders – положення повзунка.

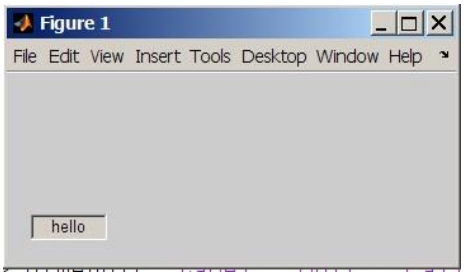


Рисунок 1.7 – Вигляд вікна з елементом уведення

Наступні рядки створюють меню **'Workspace'** з пунктами **'New Figure'**, **'Save'**, **'Quit'** (рис. 1.8).

```
f = uimenu('Label','Workspace');
uimenu(f,'Label','New Figure','Callback','figure');
uimenu(f,'Label','Save','Callback','save');
uimenu(f,'Label','Quit','Callback','exit',...
'Separator','on','Accelerator','Q');
```

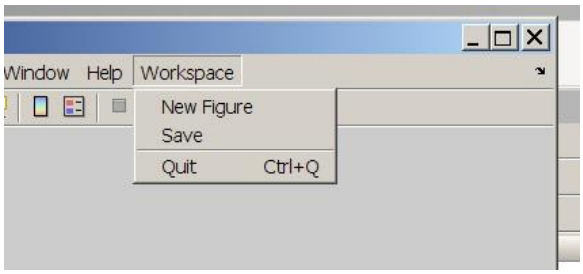


Рисунок 1.8 – Вигляд меню

Наступні рядки створюють контекстне меню.

```
cmenu = uicontextmenu;
%Define the line and associate it with the context menu
hline = plot(1:10, 'UIContextMenu', cmenu);
% Define callbacks for context menu items
cb1 = ['set(hline, ''LineStyle'', ''--'')'];
cb2 = ['set(hline, ''LineStyle'', '':'')'];
cb3 = ['set(hline, ''LineStyle'', ''-'')'];
% Define the context menu items
item1=uimenu(cmenu,'Label', 'dashed', 'Callback', cb1);
item2=uimenu(cmenu,'Label', 'dotted', 'Callback', cb2);
item3=uimenu(cmenu,     'Label',     'solid',     ...
'Callback', cb3);
```

Меню спливає при натисканні правої кнопки над лінією та має пункти **'dashed'**, **'dotted'**, **'solid'** (рис. 1.9).

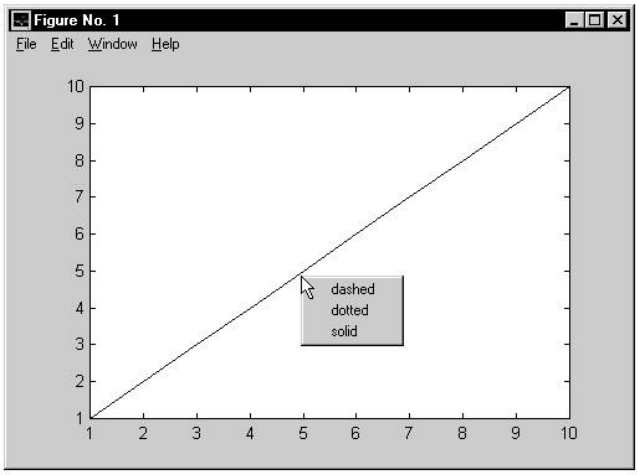


Рисунок 1.9 – Вигляд контекстного меню

При описі елементів керування (**uicontrol**) першим параметром вказується дескриптор батьківського вікна, а далі в довільному порядку перераховуються пари: ім’я властивості та значення.

Наприклад, наступні рядки створюють кнопку, при натисканні на яку буде виконуватись функція **fun** (файл **fun.m**):

```
hBut=uicontrol(hFig,'Style','pushbutton','String',…
'Кнопка', 'Position', [20 20 30 40],'Callback', 'fun');
```

Ключове слово **Style** визначає тип елемента: кнопка **pushbutton**, властивість **String** задає напис на кнопці, властивість **Position** – вектор-рядок з чотирьох чисел: перші два числа встановлюють положення кнопки відносно лівого нижнього кута графічного вікна, третє число задає ширину кнопки, а четверте – її висоту. Властивість **Callback** визначає ім’я функції **fun**, яка буде викликана при натисканні кнопки. Файл **fun.m** повинен бути створений раніше.

Діалогові вікна (вікна повідомлень)

Дружність інтерфейсу застосунка в значній мірі визначається наявністю повідомлень про можливі наслідки дій користувача. Matlab надає можливість оформити типові дії користувача у вигляді стандартних діалогових вікон Windows.

Вікно підтвердження

Деякі дії доцільно підтверджувати. Наприклад, користувач може випадково натиснути кнопку «Очистити», яка призначена для очистки осей. Для запобігання незворотних результатів перед виконанням таких дії слід

вивести діалогове вікно, в якому користувач повинен підтвердити необхідність дій.

Вікно підтвердження створюється функцією

```
questdlg('string-question', <'window label'>,...
<'button name'>,<'button name'>,<'default ...
 button name'>)
```

Аргументами функції є рядок повідомлення **string-question**, опціональний заголовок вікна **window label**, імена опціональних кнопок **button name** та ім'я кнопки за замовчанням **default button name**. Функція повертає рядок з іменем натиснутої кнопки.

Наприклад, команда

```
s=questdlg('Закрыть окно приложения?','Подтверждение ...
закрытия', 'Да', 'Нет', 'Нет');
```

виведе вікно (рис. 1.10) в якому при закритті позначкою змінна матиме значення 'Нет'.

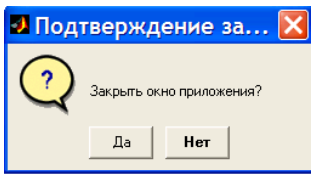


Рисунок 1.10 – Вікно підтвердження

Обробку дій користувача для вікон, які мають кілька кнопок, можна організувати після команди створення вікна по назві натиснутої користувачем кнопки:

```
switch s
    case 'Да'  delete(gcf)
    case 'Нет'  return
end
```

Вікна відкриття файлу та запису в файл

Функція

```
[fname, pname] = uigetfile(<'*.dat'>, <'title'>)
```

створює стандартне Windows діалогове вікно для обрання файлу для зчитування.

Опціональними аргументами функції є рядок з фільтром розширення файлів для відображення у вікні та тестовий рядок імені вікна **'title'.**

В разі обрання користувачем файлу функція повертає в першому результаті рядок **fname** імені, в другому **pname** – шлях до файлу. В разі закриття вікна без обрання файлу функція повертає нуль.

Функція **uiputfile** призначена для створення діалогового вікна для збереження файлу. Вхідні аргументи та вихідні результати функції **uiputfile** є такими ж, як у функції **uigetfile**.

Вікно повідомлення про помилку

Для створення діалогового вікна з повідомленням про помилку призначена функція

```
errordlg (<'errorstring'>, <'window title'>)
```

Аргументами функції є рядки з текстом повідомлення **errorstring** та заголовком вікна **window title**.

Обробка дій користувача

При виборі елемента керування виконується функція (m-файл), ім’я якої указано в властивості **Callback** елемента.

Завдання дій елемента керування можливо:

- написанням унікальних функцій оброблення для кожного елемента;
- використанням однієї функції для спільного оброблення.

У випадку використання однієї функції оброблення серед її аргументів слід передбачити ознаку, яка визначить який елемент викликає функцію. В якості такої ознаки може бути використано ім’я елемента (властивість **Tag**), покажчик елемента (**handle**) або ознака, яка призначена користувачем. В функції оброблення слід організувати перенаправлення дій в залежності від значення аргументу-ознаки, наприклад функціями **if**, **switch**.

Для програмного визначення ідентифікатора об'єкта може використовуватися функція

```
h = findobj('PropertyName',PropertyValue,...)
```

Аргументами функції є пара властивості елемента та її значення.

Наприклад, рядком

```
H_Edit=findobj('Style','edit')
```

в змінній **H_Edit** будуть збережені всі покажчики на об’єкти типу **edit**.

Рядком **H_MEdit=findobj('Tag','My_edit')**

в змінній **H_MEdit** буде збережено дескриптор об’єкта з іменем **My_edit**.

Нескладні дії можуть не програмуватися в зовнішніх файл-функціях, а визначатися безпосередньо в полі об'єкта, наприклад, елемента **edit.** Для цього зручним є застосування функції

inline('expr','arg1','arg2', ...)

Функція **inline** перетворює текстовий аргумент **expr** в функцію Matlab, значення якого може бути розраховано. Опціональні аргументи **arg** – рядки, що визначають аргументи функції.

Наприклад, запис **g=inline('sin(x)','x')** створює функцію **g(x) = sin(x),** яка може бути викликана як будь-яка функція пакета.

```
> g(1)
ans =    0.8415
```

Структура застосунку

Модуль з **GUI** повинен містити описи всіх графічних об’єктів та всі описи дій графічних елементів і дій з даними по алгоритмах користувача.

Описи дій (**callback,** функції оброблення) графічних елементів оформлюються у вигляді підпрограм-функцій користувача (**function**). Функція оброблення може бути єдиною спільною для всіх елементів графічного вікна або функції оброблення можуть записуватися для кожного елемента окремо. Особливістю використання підпрограм-функцій є те, що всі дані в них є локальними. За необхідності використання спільних структур даних в головному модулі та підпрограмах-функціях слід ввести «глобальні» змінні у всіх модулях командою **global**.

Опис вікна з **GUI** можна розмістити в головному скрипт-файлі модуля. В такому випадку скрипт вийде громіздким, що ускладнює орієнтацію в тексті модуля. Рекомендується виносити опис вікна в окремий скрипт-файл.

Версії Matlab, вищі за 7, дозволяють застосовувати підпрограми-функції без параметрів. В такому вигляді їх можна використовувати замість основного скрипту та скрипту з описом **GUI** графічного вікна.

Для прикладу напишемо скрипт для виведення графіка синьою лінією товщиною 2 функції **y(x)=sin(π·x)** для xЄ(0 10) з кроком 0.1 в поле з рамкою, з розміткою осі Y трьома значеннями "y=-l y=0 y=l", з рівномірною

розміткою осі X в діапазоні 0..10 та сіткою по осі X, з кольором фона, що співпадає з кольором графічного вікна засобами дескрипторної графіки (рис. 1.11).

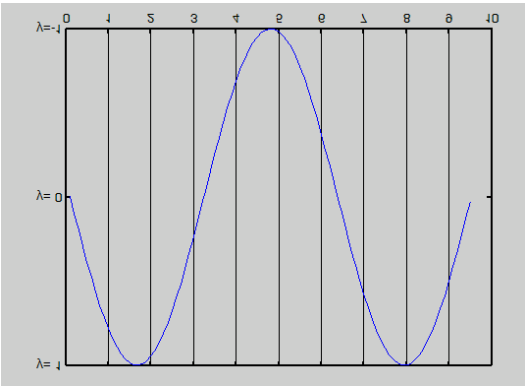


Рисунок 1.11 – Графік прикладу

```
» x=0:0.1:10;
» plot(sin(x))
» set(gca,  'Box',  'on')
» set(gca,  'LineWidth', 2)
» set(gca,  'YTick',   [-1 0 1])
» set(gca,  'YTickLabel',  ['y=-1'; 'y= 0'; 'y= 1'])
» set(gca,  'XLim',   [1 100])
» set(gca,  'XTick',   [0:10:100])
» set(gca, 'Xticklabel', [ 0;1;2;3;4;5;6;7;8;9;10])
» set(gca,  'XGrid',  'on')
» set(gca,  'GridLines',  '-')
» set(gca,  'Color',   [0.8 0.8 0.8])
```

За замовчанням графік функцією виводиться на поле білого кольору без сітки. Вісь Y має розмітку від -1 до 1 з кроком 0.2. Вісь X має розмітку від 0 до 10 з кроком 2.

Виводиться один графік, тому покажчик **gca** містить посилання на поле графіка і може бути використаний для ідентифікації дій саме з полем графіка. Осі поля графіка є елементами поля графіка. Для їхнього оформлення використано визначення властивостей поля графіка **Box** для рамки навколо графіка, **LineWidth** для товщини лінії графіка, **YTick** для координат міток по осі У, **YTickLabel** для тексту підпису осі У, **XLim** для діапазону осі X, **XTick** для координат міток по осі X, **Xticklabel** для тексту підпису осі X, **XGrid** для малювання сітки по осі

X, **GridLines** для визначення типу ліній сітки, **Color** для визначення кольору поля графіка функцією **set**.

Значення властивостей **Y(X)Tick, Color** повинні визначатися як вектори-рядки. Аргумент **Y(X)Tick** містить перелік значень, для яких потрібно нанести на вісь штрихи. Аргумент **Color** містить умовну інтенсивність червоного, зеленого, синього кольорів для отримання сірого.

Аргументами властивостей **YTickLabel, Xticklabel** є вектори-стовпці. Кожен рядок вектора містить текстові значення підписів на осях.

Приклад. Створити вікно введення даних та виведення графіка, керування виглядом для функції **y(x)=sin(x)** у відповідності до рис. 1.12 засобами дескрипторної графіки з безпосереднім описом **GUI**.

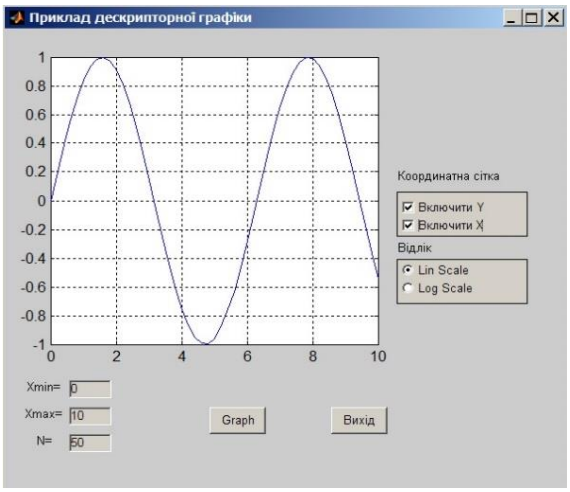


Рисунок 1. 12 – Вікно прикладу

Файл-опис вікна:

```
    function fig =priklad1_1()
    global xmin xmax step N
    %створення вікна з дескриптором f1
    f1=figure('Color',[0.8 0.8 0.8],'MenuBar','none',...
'Position',[250 100 600 550], 'Name', 'Приклад ...
дескрипторноїграфіки', 'NumberTitle','off','Tag',...
'Fig1');

    % створення поля графіки
    a1 = axes('Parent',f1, 'Units','pixels','Position', ...
    [50 200 350 320], 'Tag','Axes1', 'XColor',[0 0 0], ...
    'YColor',[0 0 0]);
    xlabel('X');
```

```
    ylabel('Y');
    title('Графік', 'FontName','Arial');

    % кнопка малювання
    a1 = uicontrol('Parent',f1, 'Callback',...
    'callfun(''press_Run'');','Position',[220 100 ...
    60 30],'String','Graph','Tag','button_Run','Style',...
'pushbutton');

    % кнопка вихід
    a1 = uicontrol('Parent',f1, 'Callback',...
    'callfun(''press_Quit'');', 'Position',[350 100 ...
    60 30],'String','Вихід', 'Tag','button_Quit');

    %кнопки сітки в рамці
    a1=uicontrol('Parent',f1, 'Position', [420 320 140 ...
50],'Style','frame', 'Tag','Frame1');

    a1 = uicontrol('Parent',f1, ...
    'Callback','callfun(''press_GridX'');', 'Position',...
    [425 325 100 15], 'String','Включити X', 'Style',  ...
'checkbox','Tag','box_GridX' );
    a1 = uicontrol('Parent',f1, 'Callback',...
    'callfun(''press_GridY'');','Position',[425 345 100 ...
15],'String','Включити Y','Style','checkbox', ...
'Tag','box_GridY');


   a1 = uicontrol('Parent',f1,'HorizontalAlignment',...
   'left','Position',[420 380 190 15],'String',...
   'Координатна сітка', 'Style','text',  'Tag',...
   'StaticText1','BackgroundColor',[0.8 0.8 0.8]);

   % кнопки масштабу в рамці
   a1 = uicontrol('Parent',f1,'Position',[420 245 140 ...
50],'Style','frame', 'Tag','Frame2');
   a1 = uicontrol('Parent',f1, 'Callback',...
   'callfun(''press_Lin'');', 'Position',[425 275 90 ...
15], 'String','Lin Scale', 'Style','radiobutton', ...
'Tag', 'Radio_Lin', 'Value',1);
   a1 = uicontrol('Parent',f1,'Callback',...
   'callfun(''press_Log'');', 'Position',[425 255 90 ...
15], 'String','Log Scale', 'Style','radiobutton', ...
'Tag','Radio_Log', 'Value',0);
   a1 = uicontrol('Parent',f1,'HorizontalAlignment',...
   'left','Position',[420 300 90 15],'String','Відлік',...
'Style','text', 'BackgroundColor',[0.8 0.8 0.8]);
```

```
    % поля уведення даних з поясненнями
    a1 = uicontrol('Parent',f1, 'Callback',...
    'callfun(''Edit_xmin'');','HorizontalAlignment',...
    'left','Position',[70 140 45 20], ...
'Style','edit', 'Tag','Edit_xmin');
    a1 = uicontrol('Parent',f1, 'Position',[20 140 45 ...
20], 'Style','text', 'string','Xmin=', ...
'BackgroundColor',[0.8 0.8 0.8]);
    a1 = uicontrol('Parent',f1, 'Callback',...
    'callfun(''Edit_xmax'');', 'HorizontalAlignment',...
    'left', 'Position',[70 110 45 20], 'Style',...
    'edit', 'Tag','Edit_xmax');
    a1 = uicontrol('Parent',f1, 'Position',[20 110 45 ...
20],Style','text', 'string','Xmax=', ...
'BackgroundColor',[0.8 0.8 0.8]);
    a1 = uicontrol('Parent',f1,'Callback',...
    'callfun(''Edit_N'');','HorizontalAlignment','left',...
    'Position',[70 80 45 20], 'Style','edit', 'Tag',...
'Edit_N');
    a1= uicontrol('Parent',f1, 'Position',[20 80 45 20],...
'Style','text', 'string','N=','BackgroundColor',...
     [0.8 0.8 0.8]);
```

Функція обробки дій (`callback`):

```
function [x,y,t,T]=callfun(event)
global xmin xmax step N;
switch event
case 'Edit_xmin'
   xmin=str2num(get(gcbo,'String'));
case 'Edit_xmax'
   xmax=str2num(get(gcbo,'String'));
case 'Edit_N'
   N=str2num(get(gcbo,'String'));
   step=(xmax-xmin)/N;
case 'press_GridX'
   if get(gcbo,'Value')
      set(gca,'Xgrid', 'on')
   else
      set(gca,'Xgrid', 'off')
   end

case 'press_GridY'
   if get(gcbo,'Value')
      set(gca,'Ygrid', 'on')
   else
```

```
set(gca,'Ygrid', 'off')
        end
     case 'press_Lin'
           h=findobj('Tag','Radio_Log');
        set(h,'Value',0);
     case 'press_Log'
          h=findobj('Tag','Radio_Lin');
        set(h,'Value',0);
     case 'press_Run'
     HStyle_Edit=findobj('Style','edit');
         if max(strcmp(get(HStyle_Edit,'String'),''))
            errordlg('Не всі дані задано','Помилка!');
         else
         h=findobj('Tag','Radio_Lin');
            if get(h, 'Value')
               x=xmin:step:xmax;

plot(x,sin(x));
             else
                x=xmin:step:xmax;
                semilogx(x,sin(x));
            end
        end
     case 'press_Quit'
        button=questdlg('Завершити роботу ?', ...
               'i1','Так','Ні','Ні');
        if strcmp(button,'Так')
           clear all
           close all
           clc
        end
     end
```

Пояснення

Модуль приклада складається з двох функцій користувача. Головна функція **priklad1_1** містить описи графічних елементів, функція **callfun** містить обробляння дій елементів. Використана одна функція обробки для всіх елементів. Для ідентифікації елемента, для якого повинно проводитися оброблення для функції **callfun** введений аргумент **event**, який є текстовим рядком. Зміст рядка визначений в описі з головної функції

та є унікальним для кожного об'єкта. Для передавання даних **xmax, xmin, step, N** ці змінні визначені глобальними.

Модуль не використовує явно покажчики об’єктів, крім покажчика об’єкта «вікно». Для визначення об’єктів задіяні системні змінні **gca**, **gcbo** та функція **findobj.**

В модулі не передбачені початкові дані. Перед виконанням дій по малюванню графіків проводиться аналіз введених в полях введення даних проводиться наступним чином. Функцією **findobj** визначається масив посилань на елементи введення типу **edit.** Функцією **get** зчитується зміст елементів з поля **tag** в масив. Функцією **strcmp** шляхом порівняння зчитаної інформації з порожнім рядком визначається чи введено якусь інформацію в поля. Якщо даних немає, то відповідному елементу масиву функція перевірки надає значення логічної одиниці. Функцією **max** визначається значення максимального елемента в масиві. Результат, який не дорівнює 0, показує, що не у всі елементи введено дані.

При оформленні діалогового вікна комп'ютерної моделі для того, щоб дослідник мав змогу легко прийняти рішення щодо обрання значень досліджуваних параметрів рекомендується дотримуватися наступних правил [6]:

- графіки мають мати такий вигляд, щоб з них можна було би зчитати значення функції з відносною похибкою в кілька процентів. Для цього координатна сітка повинна відповідати цілим числам десяткового розряду:
- графіки функцій одного аргументу, діапазони значень яких відрізняються не більше, ніж на один порядок, слід розташовувати на одному графічному полі в єдиних графічних осях:
- графіки функцій, діапазони значень яких значно відрізняються, але є додатними та мають однакову фізичну розмірність слід виводити на одне графічне поле в логарифмічному форматі;
- графіки функцій різної фізичної природи, які мають однакові аргументи слід виводити в одному вікні на різні графічних полях один під іншим, щоб однакові значення аргументу на всіх графіках розташовувалися на одній вертикалі;
- вікно комп'ютерного експерименту повинно містити детальний опис умов проведення: короткі описи об'єкта дослідження, використаної моделі, перелік вхідних та вихідних параметрів моделі та їхніх

чисельних значень, назви програми, даті проведення експерименту, дані виконавця досліджень.

Для облегшення процесу створення графічних діалогових вікон в Matlab існує середовище **GUIDE** (рис. 1.13).

Застосунок, який розроблений в середовищі **GUIDE** зберігається в двох файлах. Перший має розширення **.fig**, другий – **.m**. Перший файл містить опис розташованих у вікні об'єктах, другий файл є М-файлом з основною функцією та функціями обробки (**Callback**).

Додавання елемента інтерфейсу в редакторі **GUIDE** призводить до автоматичного створення відповідної функції обробки.

Виклик редактора **GUIDE** проводиться командою **guide**.

У вікні редактора знаходяться (рис. 1.13):

- рядок меню;
- панель інструментів керування аплікацією;
- заготовка вікна з нанесеною сіткою;
- вертикальна та горизонтальна лінійки;
- панель інструментів для вставляння елементів інтерфейсу у вікно аплікації.

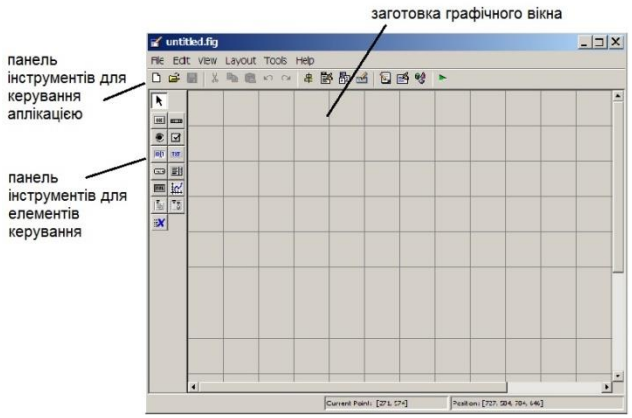


Рисунок 1.13 – Редактор діалогових вікон **GUIDE**

Користувач може використовувати в якості елементів виведення інформації та керування кнопки, перемикачі, кнопки-перемикачі, поля вибору, поля текстового введення, списки, випадаючи списки, текстові поля, декоративні рамки (групові панелі), повзунки, графіки (рис. 1.14).

Рисунок 1.14 – Панель інструментів для додавання елементів інтерфейсу

Створення елементів керування в діалоговому вікні

Для того, щоб розмістити елемент інтерфейсу в графічному вікні треба навести мишу на відповідну кнопку на панелі інструментів та клацнути мишею. Потім перевести мишу в потрібне місце вікна та клацнути мишею. Інший спосіб складається з «перетаскування» з утриманням лівою кнопкою миші кнопки з панелі інструментів в потрібне місце заготовки вікна.

Розмір, положення доданих об’єктів можуть бути змінені за допомогою миші, редактора властивостей об’єктів (**Property Inspector**), опції **Align Objects** пункту **Tool** меню чи панелі інструментів (рис. 1.15).

Будь-який об’єкт можна видалити з вікна натисканням клавіші "Delete" на клавіатурі або опції **Clear** меню.

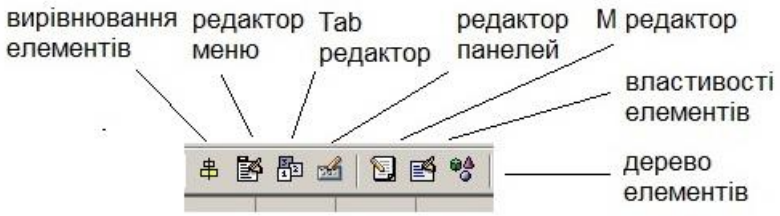


Рисунок 1.15 – Панель інструментів визначення властивостей

Редактор дозволяє проводити просте вирівнювання об’єктів за допомогою сітки, вертикальної та горизонтальної лінійок. Лінійка дозволяє розміщувати елементи інтерфейсу в позиції з будь-якими координатами в пікселях. Координати відраховуються від лівого нижнього кута заготовки вікна. Налаштування вигляду сітки та лінійок, кроку сітки, режиму прив'язки до сітки проводиться в діалоговому вікні **Grid and Rulers** (рис. 1.16). Вікно викликається пунктом **Grid and Rulers** меню **Tools.**

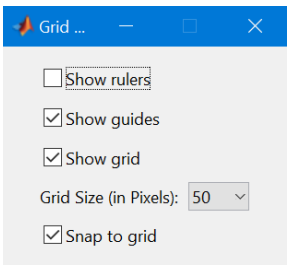


Рисунок 1.16 – Діалогове вікно **Grid and Rulers**

Мітки **Show rulers** та **Show grid** керують відображенням лінійок та сітки в вікні редактора, а випадаючий список **Grid Size** задає розмір комірок сітки. Мінімально допустимий розмір становить десять пікселів.

Перемикач **Snap to grid** керує прив'язкою переміщень до ліній сітки. Прив'язка до сітки дозволяє розташувати об'єкт та змінити його розміри тільки за умови проходження меж об'єкта по лініях сітки. Вибір мілкого кроку сітки разом з прив'язкою дозволяє швидко оформити вікно.

Плавно змінювати положення об'єкта можна за допомогою клавіш зі стрілками.

Вирівнювання об'єктів в ряд по вертикалі чи горизонталі може проводитися по обраному об'єкту або по довільній лінії.

Вирівнювання по обраному об'єкту проводиться для позначеної групи об'єктів. Вирівнювання проводиться по об'єкту, який обрано першим. Параметри вирівнювання визначаються в діалоговому вікні **Align Objects** (рис. 1.17). Вікно викликається пунктом **Align Objects** меню **Tools.**

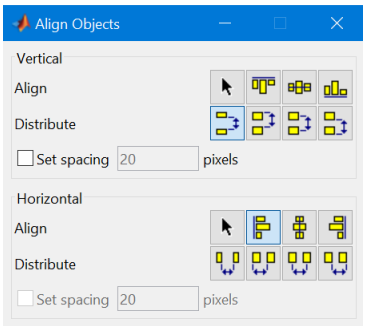


Рисунок 1.17 – Вікно вирівнювання об'єктів

Вирівнювання по лінії потребує попереднього визначення допоміжної лінії горизонтальної чи вертикальної. Слід навести курсор миші на відповідну лінійку (при наведенні курсор змінює форму на двосторонню стрілку) та, тримаючи ліву кнопку миші, витягнути в потрібне місце синю лінію. Для вирівнювання об'єктів по лінії вирівнювання слід перетягнути

потрібні об'єкти на лінію. Самі лінії вирівнювання прибираються перетягання їх мишею назад на лінійку. Видимістю ліній вирівнювання керує перемикач **Show guides** вікна **Grid and Rules**.

Прив’язка до сітки діє незалежно від стану перемикачів **Show rulers**, **Show guides**, **Show grid**.

Програмування подій

Структура функції, яка генерується редактором **GUIDE** складніше, ніж за «ручної» розробки. В ній закладено додаткові дані, структури та індивідуальні для кожного елемента **GUI** функції обробки.

Індивідуальні згенеровані **GUIDE** функції обробки мають наступний формат:

```
Function varargout= name_Callback(h,eventdata,handles,...
varargin),
```

де **name** – ім’я графічного елемента, **eventdata** – не використовується, **handles** – структура даних, в якій зберігаються покажчики графічних елементів вікна, **varargin** – змінна, яку можна використати для передачі в функцію додаткових даних.

Доступ до властивостей об'єкта через структуру **handles** проводиться функцією **get.** Об'єкт визначається своїм іменем як запис структури. Наприклад, отримати положення слайдера з іменем **slider1** можна як **get(handles.slider1,'Value');**

В структурі **handles** можна зберігати дані користувача та організувати обмін даними між функціями.

Функція, яка передає дані, повинна записати дані в визначене поле та зберегти структуру командою **guidata**.

Наприклад, можна зберегти масив [1 2 3 4] в полі **dat** структури **handles**, а потім використати його в іншій функції:

```
handles.dat =  [1 2 3 4];
guidata(gcbo, handles);
max(handles.dat);
```

Обробка списків та радіорядків

Дії по обробленню випадаючого списку та радіорядків має особливості.

Властивість **value** випадаючих списків містить номер обраного в списку рядка. Рядки в списку нумеруються з одиниці. В списках може бути виділено кілька елементів. Для множинного вибору властивість **value** містить вектор номерів обраних елементів. Можливість множинного вибору визначається значеннями властивостей **Max** и **Min**. Якщо різниця **Max – Min** більше одиниці, то можна виділяти кілька рядків.

Оброблення дій радіорядків проводиться не для кожної радіокнопки, а для всього рядка. Властивість **value** радіорядка містить ім'я **tag** активованої радіокнопки у вигляді текстового рядка. Подальше розгалуження алгоритму потребує логічних функцій на зразок **switch** з перевіркою співпадіння прочитаного змісту властивості value з іменами радіокнопок графічного вікна.

Застосування редактора **GUIDE**:

- суттєво зменшує обсяг дій при створенні опису GUI, особливо в операціях визначення положення та вирівнювання елементів;
- знижує вимоги до кваліфікації користувача як програміста;
- полегшує передавання даних між модулями;
- не дозволяє редагувати графічні елементи без застосування редактора;
- обмежує міграцію застосунків між версіями Matlab.

Застосування «ручного» програмування:

- збільшує обсяг дій при створенні опису GUI;
- вимагає підвищеної кваліфікації користувача як програміста;
- ускладнює передавання даних між модулями;
- забезпечує вільний вибір типів підпрограм застосунку;
- дозволяє редагувати графічні елементи без застосування редактора;
- не обмежує міграцію застосунків між версіями Matlab.

Контрольні запитання

Що називають моделюванням?

Що таке модель?

Чим відрізняються системний та індуктивний підходи до моделювання?

Які види входять до предметних моделей?

Що таке математична модель?

В чому достоїнства та недоліки аналітичних моделей?

В чому достоїнства та недоліки імітаційних моделей?

Які особливості мають інженерні задачі?

З яких етапів складається розв'язання інженерної задачі?

Які є складові похибки математичної моделі?

Які математичні моделі є детермінованими?

Які математичні моделі є стохастичними?

Які математичні моделі є нелінійними?

Які математичні моделі є нестаціонарними?

Які математичні моделі називається функціональними, а які - структурними?

Яка форма оформлення комп'ютерного експерименту вважається найефективнішою?

З яких етапів складається побудова об'єктної моделі Matlab?

Якими засобами створюється об'єктна модель Matlab?

Що є основою дескрипторної графіки Matlab?

Що таке покажчик в дескрипторній графіці Matlab?

Які типи вікон є в дескрипторній графіці Matlab?

Як створюються об'єкти в дескрипторній графіці Matlab?

Які механізми призначення властивостей є в дескрипторній графіці Matlab?

Які типи вікон повідомлень є в дескрипторній графіці Matlab?

Якими способами можна обробляти дії користувача в дескрипторній графіці Matlab?

Що повинен містити модуль Matlab?

Для чого призначено середовище **GUIDE?**

Які елементи входять до функцій обробки **GUIDE?**

Яку інформацію. має містити вікно комп'ютерного експерименту?

2. ЗАСАДИ КОМП'ЮТЕРНИХ ОБЧИСЛЕНЬ

2.1. Похибки комп'ютерних розрахунків

Результат обчислення математичного виразу або застосування чисельного методу, зазвичай є наближеним, тобто містить якусь **похибку**.

Джерелами погрішності є:

- невідповідність математичної постановки задачі досліджуваному реальному явищу – похибка постановки задачі;
- похибка відправних даних – початкова або неусувна похибка;
- похибка чисельного методу розв'язання – методична похибка;
- похибки заокруглення та розрахунків: похибка, пов'язана з обриванням нескінченних процесів (наприклад, ряду) – залишкова похибка, похибки округлення в комп'ютерних представленнях чисел, похибки розрахунків з наближеними значеннями.

Наближеним числом x' називається число, що незначно відрізняється від точного числа X і яке заміняє його в обчисленнях.

Наприклад, комісія при перерахуванні банком коштів складає 2.5 %. Тобто за перерахунок 247 грн сплата складає 6.175 грн. Число 6,175 є точним, але банк виставить до оплати наближене значення 6.18 грн.

В виразах «546 сторінок в книзі», «9 поверхів в будівлі», «2021 рік» числа 546, 9, 2019 є точними.

У фразах «довжина деталі 20 мм», «температура тіла 36.6 °С», «тривалість експерименту 21 хв», «словник на 25000 слів», «в добі 24 години», числа 20, 36.6, 21, 25000, 24 є наближеними.

Якщо $x'<X$, то x' є наближенням з **недостачею**, якщо $x'>X$ – з **надлишком**.

Наближене число, задане тільки своїм значенням, практичного змісту не має, оскільки невідомим є ступінь його наближення до точного. Наближена рівність $x'\approx X$ має зміст тільки тоді, коли відома похибка, з якою обчислене x'.

Для визначення ступеню точності наближення користуються поняттям **абсолютної** похибки.

Абсолютна величина різниці між **x'** та **X** називається **абсолютною похибкою**

$$\Delta=|x'-X|.$$

Абсолютна похибка, як правило, практичного значення немає, бо зазвичай точне значення **X** невідоме. Тому частіше використовується поняття **граничної абсолютної похибки**

$$-\Delta x\le|x'-X|\le\Delta x,\quad X=x'\pm\Delta x.$$

Останній вираз означає, що число **x'** є наближеним значенням величини **X** з похибкою **Δx**. Під **граничною абсолютною похибкою** розуміють будь яке число, яке є не меншим за **абсолютну похибку** $\Delta\le\Delta x$.

Наприклад, наближене значення $x=3.14$ квадратури кола має граничну абсолютну похибку $\Delta x=0.01,\ 3.14<\pi<3.15,|x-\pi|<0.01$.

Невідомий точний розв'язок задачі знаходиться від на відстані не більше, ніж Δ від значення наближеного розв’язку.

Оскільки **гранична абсолютна похибка** використовується для характеристики порядку точності наближеного значення числа, то на практиці обмежуються однією-двома значущими цифрами. При цьому заокруглення проводиться лише в сторону збільшення [20; 21].

Гранична абсолютна похибка є розмірною величиною, або іменованим числом. її розмірність така ж, як і в самого наближення.

Значення **абсолютної похибки** чи **граничної абсолютної похибки** НЕ ДОЗВОЛЯЮТЬ повністю характеризувати якість вимірювання чи точність обчислень [6].

Наприклад, для двох наближених значень $x1=222.2\pm0,1$ та $x2=5.5\pm0,1$ значення **граничної абсолютної похибки** є однаковими та складають 0.1. При цьому якість значення **x1** вища за якість значення **x2**.

Для оцінки якості вимірювання чи розрахунків використовується **відносна похибка**.

Відносна похибка **δ** наближеного числа **x'** є відношенням **абсолютної похибки** цього числа до модуля точного числа.

$$\delta=|\Delta|/|X|.$$

Граничною відносною похибкою називають відношення граничної абсолютної похибки до модуля наближеного числа. Граничною відносною похибкою є всяке число, яке є не меншим за значення відносної похибки.

$$\delta x=|\Delta x|/|x'|,\quad \delta\le\delta x,\quad \Delta x=\delta x\cdot|x'|.$$

Гранична абсолютна похибка є добутком множення **відносної граничної похибки** на значення наближеного числа.

Наприклад, для наближених чисел **x, x1** та **x2** гранична відносна похибка складає $\delta x=0.01/3.14=0.32\ \%,\delta x1=0.1/222.2=0.045\ \%,\ \delta x2=0.1/5.5=1.8\ \%.$

Значущі цифри. Правильні та сумнівні цифри

Запис наближеного числа через **граничну абсолютну похибку** не завжди зручна. В літературі та документації використовується спосіб запису, який дозволяє за десятковим записом числа визначити **граничну абсолютну похибку** наближення.

Кожне число x, записане в десятковій системі числення, можна представити в наступному вигляді $x = M\cdot 10^p$, де **M** – мантиса числа, **p** – порядок числа. Наведений запис називається **плаваючою формою** запису числа. **Плаваюча форма** для одного й того ж числа може мати різний вигляд. Наприклад, $12345=12.345\cdot 10^3=123.45\cdot 10^2=123450\cdot 10^{-1}.$

Якщо для множника **M** виконується нерівність $0<M<1$, то форма називається **нормальною**. Запис числа в **нормальній** формі **не є єдиним**. Наприклад, $12.345=0.12345\cdot 10^2=0.012345\cdot 10^3$.

Нормальна форма запису числа, в мантисі якої першою після коми є відмінна від нуля цифра, називається **нормалізованою**. Представлення числа у **нормалізованій** формі **є єдиним**.

Якщо для множника **M** виконується нерівність $1<M<10$, то форма називається **стандартною** формою запису. Представлення числа в **стандартній** формі **є єдиним**. Наприклад, $12345=1.345\cdot 10^2$.

Значущими цифрами наближеного числа називають всі його цифри, починаючи з першої зліва ненульової цифри. Крайні справа нулі є значущими тільки в тому випадку, коли відомо, що одиниць відповідного розряду в даному числі немає.

Наприклад, наближені числа **219.5, 0.0213, 0.2205, 0.001796, 0.8000** мають по 4 **значущі** цифри. В числі **0.8000** крайні справа нулі означають відсутність одиниць відповідних розрядів, тобто важливим є саме значення «нуль цілих вісім тисяч десятитисячних».

В числі **410±5** є дві **значущі** цифри 4 і 1. Останній нуль не є **значущим**, бо він замінює невідому цифру одиниць.

Серед **значущих** розрізняють **правильні** та **сумнівні** цифри.

В позиційній системі числення число записується у вигляді скінченого дробу:

$$x=a_mR_m+a_{m-1}R_{m-1}+...+a_0R^0, \quad (a_m<>0),$$

де a_i – цифри, R – основа числення (для десяткової системи $R=10$).

Степінь основи визначає **розряд** відповідної цифри.

Наприклад, число $123.45=1\cdot10^2+2\cdot*10^1+3\cdot10^0+4\cdot10^{-1}+5\cdot10^{-2}$ має 5 розрядів: старший – сотні, молодший – соті. Цифрі 1 відповідає другий розряд (одиниця розряду 100), 2 – першій (1) тощо.

Значуща цифра наближеного значення числа називається **правильною** в **суворому** розумінні, якщо абсолютна похибка наближення не перевищує половини одиниці того розряду, якому належить цифра **a**.

Значуща цифра наближеного значення числа називається **правильною** в **широкому** розумінні, якщо абсолютна похибка наближення не перевищує одиниці того розряду, якому належить цифра **a**.

Цифри в записі наближеного числа, які не є **правильними**, називаються **сумнівними**, і на письмі повинні **підкреслюватися**. Наприклад, **1.2345**.

Всі **значущі** цифри наближеного числа, що стоять зліва від **правильної** є **правильними**. Всі **значущі** цифри наближеного числа, що стоять справа від **сумнівної** є **сумнівними**.

Наприклад, для наближеного числа $x=1.743\pm0.01$ цифра 3 відповідає розряду тисячних (10^{-3}). Одиниця цього розряду становить 0.001. Значення одиниці цього розряду менше за граничну абсолютну похибку ($0.001 < 0.01$), тому 3 є сумнівною цифрою в широкому сенсі. Значення половини одиниці

цього розряду менше за граничну абсолютну похибку (0.0005 < 0.01), тому 3 є сумнівною цифрою в суворому сенсі.

Цифра 4 відповідає розряду сотих (10^{-2}). Одиниця цього розряду становить 0.01. Значення одиниці цього розряду дорівнює граничній абсолютній похибці (0.01 = 0.01), тому 4 є правильною цифрою в широкому сенсі. Значення половини одиниці цього розряду менше за граничну абсолютну похибку (0.005 < 0.01), тому 4 є сумнівною цифрою в суворому сенсі.

Цифра 7 відповідає розряду десятих (10^{-1}). В широкому сенсі ця цифра є правильною, бо стоїть зліва від правильної цифри 4. Значення половини одиниці цього розряду більше за граничну абсолютну похибку (0.05 > 0.01), тому 7 є правильною цифрою в суворому сенсі.

Запис може бути таким **1.743** для широкого, **1.743** для суворого сенсів.

Термін «**правильна** цифра» не слід сприймати буквально, оскільки **правильна** цифра може і не співпадати з відповідною цифрою точного значення числа. Наприклад, для наближеного значення числа π **x=3.142±0.0001** – всі цифри числа 3.142 є правильними в суворому сенсі, проте цифра 2 не співпадає з відповідною цифрою точного числа.

Якщо наближене число закінчується **правильними** значущими нулями, які стоять після коми, то ці нулі не відкидаються. Наприклад, для **x=7.1±0.0005** слід писати **x=7.100**, для **x=123±0.005**, **x=123.00**.

Згідно правилу, запропонованому А. Н. Криловим, наближене число потрібно записувати так, щоб усі **значущі** цифри у запису числа були **вірними**, а перша цифра з відкинутої частини відносилася до **сумнівних** [3]. Так, число **0.884±0.004**, має бути записано як **0.88**.

Округлення чисел

Округлення зводиться до заміни точного числа наближеним або наближеного числа іншим наближеним, менш точним. При **округленні** частина значущих цифр відкидається, або при необхідності замінюється нулями.

Якщо результат **округлення** позначити через **x'**, то абсолютна величина різниці |**X − x'**| називається **похибкою округлення**. Число **x'** вибирають так, щоб ця похибка була найменшою.

Для проведення округлення використовують наступні правила округлення [20; 21]:

Якщо перша зліва із цифр, які відкидаються,

- більша, ніж 5 або дорівнює 5 і за нею стоять відмінні від нуля цифри то остання із залишених цифр збільшується на одиницю;
- менша, ніж 5, то остання із залишених цифр не змінюється;
- дорівнює 5 і за нею стоїть нуль, то остання із залишених цифр збільшується на одиницю, якщо вона непарна, і залишається без змін, якщо вона парна. Це правило називається **правилом парної цифри.** Застосування цього правила до одного числа не призводить до збільшення точності обчислень. Проте при багаторазових заокругленнях надлишкові числа зустрічаються приблизно так само часто, як числа з недостачею. Тому має місце взаємна компенсація похибок і результат виявляється більш точним.

Гранична абсолютна похибка округлення не перевищує половини одиниці того розряду, якому належить остання залишена цифра.

Похибки арифметичних обчислень

Гранична абсолютна похибка **алгебраїчної суми** наближених чисел дорівнює сумі граничних абсолютних похибок цих чисел.

$$\Delta_\Sigma = \mp\Delta x_1 \mp \Delta x_2 \ldots \mp \Delta x_N \le \Delta x_1 + \Delta x_2 + \cdots + \Delta x_N = \sum_{i=1}^{N} \Delta x_i.$$

Гранична абсолютна похибка суми не може бути меншою граничної абсолютної похибки найменш точного із доданків.

При округленні N доданків суми m-го десяткового знаку абсолютна похибка округлення суми в самому гіршому випадку не перевищує [21]:

$$\Delta_{\Sigma max} = \frac{N}{2} 10^{-m},$$

де **N** – кількість доданків.

Відносна гранична похибка суми двох доданків виражається як

$$\delta_{a+b} = \frac{\Delta_a + \Delta_b}{|a+b|} = \delta_a \frac{|a|}{|a+b|} + \delta_b \frac{|b|}{|a+b|}.$$

Якщо доданки мають один знак, то гранична **відносна** похибка їх **суми** не перевищує найбільшої з граничних відносних похибок доданків.

$$\delta_\Sigma = \frac{\Delta_\Sigma}{X_1 + X_2 + \cdots + X_N} = \frac{\Delta x_1 + \Delta x_2 + \cdots + \Delta x_N}{X_1 + X_2 + \cdots + X_N} = \frac{X_1\delta_1 + X_2\delta_2 + \cdots + X_N\delta_N}{X_1 + X_2 + \cdots + X_N}$$

$$= \{\delta = \max(\delta_i)\} \le \frac{\delta(X_1 + X_2 + \cdots + X_N)}{X_1 + X_2 + \cdots + X_N} = \delta.$$

Додавання величин протилежного знаку (або віднімання величин однакового знаку) приводить до збільшення відносної похибки результату у порівнянні з найбільшою з відносних похибок доданків. Особливо небезпечним є віднімання дуже близьких величин. У цьому випадку відносна похибка результату може сягати неприпустимих величин [6; 21].

Гранична **відносна похибка добутку** наближених чисел дорівнює сумі граничних відносних похибок співмножників.

$$(a \mp \Delta a)(b \mp \Delta b) = ab \mp a\Delta b \mp b\Delta a \mp \Delta a\Delta b \le ab \mp (a\Delta b + b\Delta a + \Delta a\Delta b),$$

$$\Delta = a\Delta b + b\Delta a + \Delta a\Delta b \approx a\Delta b + b\Delta a,$$

$$\delta \approx \frac{\Delta}{ab} = \frac{a\Delta b + b\Delta a}{ab} = \frac{\Delta b}{b} + \frac{\Delta a}{a} = \delta_a + \delta_b.$$

Абсолютна гранична похибка добутку виражається через відносну граничну похибку як

$$\Delta = |ab| \cdot \delta.$$

Гранична **відносна похибка m-го степеня** числа в **m** разів більша граничної **відносної похибки самого числа**.

При **множенні наближеного числа на точний множник k** відносна гранична похибка не змінюється, а абсолютна гранична похибка збільшується в **k** разів.

Гранична **відносна похибка частки** наближених чисел дорівнює сумі граничних відносних похибок діленого та дільника.

Розглянемо логарифм частки $u = \frac{a}{b}$:

$$\ln(u) = \ln(a) - \ln(b)\ \ \Delta\ln(x) \approx \frac{d(\ln(x))}{dx}dx = \frac{dx}{x} \approx \frac{\Delta x}{x},$$

$$\Delta\ln(u) = \frac{\Delta a}{a} + \frac{\Delta b}{b} \approx \frac{\Delta u}{u},$$

$$\delta \approx \left|\frac{\Delta u}{u}\right| \leq \left|\frac{\Delta a}{a}\right| + \left|\frac{\Delta b}{b}\right| = \delta a + \delta b.$$

Якщо кількість правильних знаків в найменш точному із співмножників, дільника чи діленого – **m**, то кількість правильних знаків в добутку/ частці на одну або дві одиниці менше **m**.

Щоб провести розрахунки з числами різної абсолютної точності потрібно [20; 21]:

- виділити числа, які мають найменшу кількість правильних цифр;
- інші числа округлити за зразком виділених, зберігаючи один (два) запасних десяткових знаків;
- провести розрахунки чисел, враховуючи всі збережені знаки;
- одержаний результат округлити на один (два) розряд. При піднесенні до степеня (добуванні кореня) у результаті необхідно зберігати стільки значущих цифр, скільки правильних значущих цифр має основа степеня (підкореневе число).

Якщо дані можна брати із довільною точністю, то для одержання результату з **k** правильними цифрами вихідні дані потрібно брати із такою кількістю цифр, яка забезпечує **k +1** правильну цифру у результаті.

Похибки значень функцій [21]

Прямою задачею є визначення **абсолютної похибки** диференційованої **функції f**, що залежить від **N** змінних за відомими $X_1, X_2, \ldots, X_N$ – точними значеннями змінних, $x_1', x_2', \ldots, x_N'$ – наближеними значеннями змінних, $\Delta_i = |X_i - x_i'|$ – їх абсолютними похибками.

Абсолютна похибка значення функції в точці ($x_1, x_2, \ldots, x_N$) – це її приріст $\Delta_f = |f(X_1, X_2, \ldots, X_N) - f(x_1', x_2', \ldots, x_N')|$.

Для $\Delta_i << x_i'$ приріст функції можна замінити її диференціалом та отримати вираз для абсолютної похибки

$$\Delta f = |f(X_1, X_2, \dots, X_N) - f(x_1, x_2, \dots, x_N)| \approx |df(x_1', x_2', \dots, x_N')|$$
$$\approx |\sum_{i=1}^{N} \frac{df}{dx_i} dx_i| \leq \sum_{i=1}^{N} \left|\frac{df}{dx_i}\right| \Delta x_i.$$

З виразу абсолютної похибки можна отримати оцінку **граничної відносної похибки функції**:

$$\delta_f = \frac{\Delta_f}{|f|} = \sum_{i=1}^{N} \frac{|f_{x_i}'|}{|f|} \Delta x_i = \sum_{i=1}^{N} \left|\frac{d(\ln(f))}{dx_i}\right| \Delta x_i = \sum_{i=1}^{N} \left|\frac{d(\ln(f))}{dx_i}\right| |x_i| \delta x_i.$$

Похибки найуживаніших математичних операцій наведено в таблиці 2.1.

Таблиця 2.1. Похибки математичних операцій [2]

Операція	Δ	δ	Операція	Δ	δ
$x + y$	$\Delta_x + \Delta_y$	$\frac{\|x\|}{\|x+y\|}\delta_x + \frac{\|y\|}{\|x+y\|}\delta_y$	x/y	$\frac{x\Delta_y + y\Delta_x}{y^2}$	$\delta_x + \delta_y$
$x - y$	$\Delta_x + \Delta_y$	$\frac{\|x\|}{\|x-y\|}\delta_x + \frac{\|y\|}{\|x-y\|}\delta_y$	x^n	$nx^{n-1}\Delta_x$	$n\delta_x$
$x \cdot y$	$x\Delta_y + y\Delta_x$	$\delta_x + \delta_y$	$\sqrt[n]{x}$	$\frac{\sqrt[n]{x}}{nx}\Delta_x$	$\frac{\delta_x}{n}$

Обернена задача теорії похибок

Обернена задача полягає в визначені значень **абсолютних** похибок **аргументів** функції $\mathbf{f(X_1, X_2, \dots, X_N)}$, за яких абсолютна похибка функції не перевищуватиме заданої величини $\mathbf{\Delta f}$.

Ця задача математично не визначена, оскільки задану граничну похибку $\mathbf{\Delta f}$ функції $\mathbf{f(X_1, X_2, \dots, X_N)}$ можуть визначити різні абсолютні похибки $\mathbf{\Delta x_i}$ її аргументів.

Для розв'язання оберненої задачі застосовують методи **рівних впливів**, **рівних абсолютних граничних похибок**, **рівних відносних граничних похибок**.

Принцип рівних впливів. За цим принципом всі частинні диференціали $|\frac{df}{dx_i}|\Delta x_i$ однаково впливають на величину загальної похибки $\mathbf{\Delta f}$ функції , тобто мають однакові значення:

$$\left|\frac{df}{dx_1}\right|\Delta x_1 = \left|\frac{df}{dx_2}\right|\Delta x_2 = \cdots = \left|\frac{df}{dx_N}\right|\Delta x_N = \frac{\Delta f}{N}.$$

Звідки

$$\Delta x_i = \frac{\Delta f}{N|\frac{df}{dx_i}|}.$$

Принцип рівності граничних абсолютних похибок. За цим принципом всі граничні абсолютні похибки аргументів функції мають однакові значення $\Delta x_i = const$.

$$\Delta f = \Delta x \sum_{i=1}^{N}\left|\frac{df}{dx_i}\right|, \qquad \Delta x = \frac{\Delta f}{\sum_{i=1}^{N}|\frac{df}{dx_i}|}.$$

Принцип рівності відносних абсолютних похибок. За цим принципом всі відносні похибки аргументів функції мають однакові значення $\frac{\Delta x_i}{|x_i|} = const = \delta$.

$$\Delta f = \sum_{i=1}^{N}|\frac{df}{dx_i}|\Delta x_i = \left\{\frac{\Delta x_i}{|x_i|} = \delta,\ \ \Delta x_i = \delta|x_i|\right\} = \delta\sum_{i=1}^{N}\left|\frac{df}{dx_i}\right||x_i|,$$

$$\delta = \frac{\Delta f}{\sum_{i=1}^{N}|\frac{df}{dx_i}||x_i|}.$$

З урахуванням наведеного

$$\Delta x_i = \frac{\Delta f|x_i|}{\sum_{i=1}^{N}|\frac{df}{dx_i}||x_i|}.$$

Похибка округлення комп'ютерних розрахунків

Скінчений обсяг комірок комп'ютерної пам’яті (розрядність сітки, кількість знаків мантиси) призводить до того, що дані та результати обчислень стають наближеними за рахунок округлення знаків числа, які не вміщуються в комірку пам'яті, та викликають обчислювальну похибку округлення.

При розв'язанні обчислювальних задач зазвичай використовують подання чисел у формі з плаваючою крапкою (комою).

Число **a** у формі з плаваючою крапкою за стандартом IEEE 754:2008 подається у вигляді:

$$a = sign(a)\cdot M\cdot r^{p},$$

де **sign(a)** – знак числа a; **M** – значущі цифри числа **a**; **r** – основа системи числення; **p** – порядок числа **a**.

При цьому виконується **умова нормування,** коли перша цифра в мантисі є відмінною від нуля (**стандартизована форма**).

Для двійкової системи числення (**r=2**). найбільш популярні типи чисел із плаваючою крапкою: числа звичайної точності **float**, **single** (4 байти) і числа подвійної точності **double** (8 байтів) мають наступну структуру:

Тип	Знак	Мантиса	Знак	Порядок
float	1	23	1	7
double	1	52	1	10

Мантиса зберігає **k=M·lg(2)≈0.3M** десяткових знаків.

Для чисел звичайної точності к≈7. Діапазон порядків становить $\pm 2^7 = \pm 127$. Для довідки, діапазон **float** мови С становить ±38 , Fortran ±32, Pascal – 45…+38.

Точне число "**a**" у вигляді нескінченого дробу

$$a = \pm \underbrace{2^{p}}_{\text{порядок}} \underbrace{\left(\frac{a_1}{2} + \frac{a_2}{2^2} + \cdots + \frac{a_t}{2^t} + \frac{a_{t+1}}{2^{t+1}} + \cdots \right)}_{},$$

неможливо обробити в комп'ютері. Воно піддається округленню, тобто замінюється близьким числом $\tilde{a}$ з розрядністю мантиси **t**.

$$\tilde{a} = \pm 2^{p} \left(\frac{a_1}{2} + \frac{a_2}{2^2} + \cdots + \frac{a_t}{2^t} \right). \qquad (2.1)$$

де **ai = 0/1, (i=1,2,...,t)** – цифри мантиси; **t** – розрядність мантиси (23 або 52 відповідно до типу представлення дійсного числа).

Просте округлення проводиться відкидання всіх розрядів числа, що виходять за межею розрядної сітки. При цьому, буде отримане округлене число з абсолютною похибкою $\tilde{a} - a$

$$\tilde{a} - a = \pm 2^{p}\left(\frac{a_{t+1}}{2^{t+1}} + \frac{a_{t+2}}{2^{t+2}} + \cdots\right).$$

Найбільша похибка буде у випадку, коли $a_{t+1}=1$, $a_{t+2}=1$. Тоді абсолютна похибка округлення

$$\left|\tilde{a} - a\right| \le 2^{p}\,\frac{1}{2^{t+1}}\underbrace{\left(1 + \frac{1}{2} + \frac{1}{2^{2}} + \cdots\right)}_{=2} = 2^{p-t}.$$

З умови нормування M >1/2, а значить завжди $a_1=1$. Тоді з (2.1) отримуємо, що $|a|>2^{p}\cdot(1/2)=2^{p-1}$ і тому $|a-a`|/|a|<2^{-t+1}$ тобто **відносна похибка** заокруглення в комп'ютерних комірках дорівнює 2^{-t+1}.

В сучасних комп'ютерних застосунках застосовують складніші методи округлення, тому відносна похибка подання чисел в пам'яті комп'ютера дорівнює:

$$\frac{\left|\tilde{a} - a\right|}{\left|a\right|} \le 2^{-t},$$

тобто точність подання чисел визначається розрядністю мантиси t.

Наближено подане в комп'ютері число ã можна записати в вигляді $\tilde{a} = a \cdot (1 \mp \varepsilon)$, де $|\epsilon|<2^{-t}$.

Тому відносну похибку подання чисел 2^{-t} називають «**машинним епсилоном**».

2.2. Чисельні методи

Математична модель є множиною виразів, які можуть вміщувати інтегральні, диференціальні, функціональні перетворення, систем рівнянь, в тому числі нелінійних та інтегро-диференціальних.

Аналітичні методи передбачають отримання розв'язку задачі у вигляді аналітичних виразів. Їх **перевагами** є запис розв'язку у загальному вигляді, та висока точність.

Основний **недолік аналітичних методів –** неуніверсальність, бо тільки невелика частина математичних задач може бути розв'язана аналітично, для багатьох систем невідомі аналітичні залежності для опису вхідних та системних характеристик. Результат може не мати запису в елементарних функціях та виражається через спеціальні (гамма-функція, функції Беселя тощо), нескінченні ряди і взагалі не мати аналітичного розв'язку.

Крім того отримання результату вимагає високої математичної кваліфікації дослідника.

Для інженерних застосунків потребується доведення аналітичного виразу до кількісної відповіді, яка визначає значення параметра, характеристики системи, що моделюється. Отримання таких відповідей базується на ч**исельних методах,** що дозволяють звести розв'язування задачі до виконання скінченного числа арифметичних і логічних дій з числами. При цьому розв'язок визначається як набір чисел. Їх **перевагами** є абсолютна універсальність, бо теоретично будь-які математичні дії можуть бути зведені до виконання скінченного числа арифметичних і логічних дій.

Чисельні методи є основним апаратом розв'язання математичних задач, а їх вага тільки збільшуватиметься у міру вдосконалення комп'ютерної техніки.

Розрізняють чисельні методи двох типів: **прямі** та **ітераційні.**

В **прямих** методах розв'язок задачі досягається за скінченну кількість **кроків** методу. **Прямі** методи називають **точними**, бо вони дозволяють здобути розв'язок задачі без **методичних** похибок. До таких методів можна віднести метод Гауса для розв'язування системи лінійних алгебраїчних рівнянь (СЛАР), симплекс-метод розв'язування задачі лінійного програмування тощо.

В **ітераційних** методах виконується заздалегідь невідома кількість кроків **ітерацій** методу до отримання наближеного розв'язку із заданою точністю.

Ітераційні методи дозволяють здобути наближений розв'язок задачі ТІЛЬКИ з певною заданою похибкою. Такі методи називають наближеними.

Крок ітерації [22]

Ітерація – це один крок повторення сукупності операцій, які містять процедури обраного методу чи моделі. На кожному кроці використанням низки формул новий уточнений наближений розв'язок x_{k+i} обчислюється через попередній x_k, тобто $x_{k+1} = F(x_k)$.

Нехай x^* – розв'язок задачі. Тоді ітераційний метод будує так звану **ітераційну послідовність** $\{x_k\}$ $k=0...N$ наближень розв'язку, при цьому значення x_k повинно ближче наближатися до x^* зі збільшенням k.

Алгоритм ітераційного методу в найзагальнішому вигляді має таку схему:

1. Задати **початкове наближення** розв'язку (на основі апріорних знань про задачу) x_1 для першого кроку ітерації $k=1$.
2. Запустити наступний крок ітерації $k+1$. Обчислити наступне наближення $x_{k+i}=F(x_k)$, де F є сукупністю операцій методу.
3. Перевірити **критерій останову,** чи є отримане наближення розв'язку x_{k+1} достатньо близьким x^*. Якщо цього немає, то перейти до наступної ітерації, тобто до пункту 2.

Вигляд **критерію останову** (тобто припинення обчислень за ітераційним методом) залежить від типу розв'язуваної математичної задачі. Для обчислення визначеного інтегралу x – це значення інтегралу, для розв'язання рівнянь – корінь тощо.

Записується критерій останову з точністю ε_x у вигляді

$$||x_{k+1} - x_k|| \leq \varepsilon_x,$$

де $||\ ||$ – евклідова норма.

Чисельний метод називається **таким, що збігається,** якщо наближення x_k прямує до розв'язку x^* зі степенем n.

Для збіжності метода потрібно, щоб відстань між знайденим наближенням $\mathbf{x_{к+1}}$ та розв'язком $\mathbf{x^*}$ $\Delta_{k+1} = \|x_{k+1} - x*\|$ на к+1 кроці була меншою за відстань $\Delta_k = \|x_k - x*\|$ на k-му кроці ітераційного процесу. Чим менше відношення $\mathbf{\Delta к+1 / \Delta к}$, тим з більшою швидкістю наближення методу збігаються до розв'язку.

Метод має збіжність степеня n, якщо існує таке число 0<C<1, що

$$\lim_{k\to\infty} \left(\frac{\Delta_{k+1}}{\Delta_k}\right)^n = C.$$

Послідовність $\mathbf{\{x_k\}}$ **лінійно** збігається (лінійна збіжність) до розв'язку $\mathbf{x^*}$, якщо існує таке число $C \in (0,1)$ для всіх k, для якого вірна нерівність

$$\left\|\frac{\Delta_{k+1}}{\Delta_k}\right\| \le C, k \to \infty.$$

Послідовність $\mathbf{\{x_k\}}$ **надлінійно** збігається (надлінійна збіжність) до розв'язку $\mathbf{x^*}$, якщо існує така послідовність чисел $C_k \in (0,1)$ для всіх k, для якого вірна нерівність

$$\left\|\frac{\Delta_{k+1}}{\Delta_k}\right\| \le C_k, k \to \infty.$$

Послідовність $\mathbf{\{x_k\}}$ **квадратично** збігається до розв'язку $\mathbf{x^*}$, якщо існує таке число $C \in (0,1)$ для всіх k, для якого вірна нерівність

$$\left\|\frac{\Delta_{k+1}}{\Delta_k}\right\|^2 \le C_k, k \to \infty.$$

Методи, які не збігаються, не мають користі з прикладної точки зору. Тому одним з найважливіших етапів при введенні нового чисельного методу є теоретичне доведення його збіжності, тобто формулювання умов, за яких метод гарантовано збігається.

Особливостями всіх чисельних методів є те, що вони [7]:

- потребують проведення великої кількості рутинних арифметичних обчислень за допомогою співвідношень, що використовуються для організації **ітерацій** тобто повторюваних циклів обчислень зі зміненими початковим умовами для поліпшення результату;
- направлені на знаходження **локального** розв'язку задачі;

- результат залежить від близькості початкового наближення (або декількох наближень), необхідного для початку обчислень до розв'язку.

Контрольні запитання

Назвіть джерела погрішностей комп’ютерних розрахунків.

Яке значення називається наближенням з недостачею?

Яке значення називається наближенням з надлишком?

Що характеризує абсолютна похибка?

Що таке абсолютна гранична похибка?

Що характеризує відносна похибка?

За якими формами запис числа є єдиним?

В чому різниця між нормальною та нормалізованою формами запису числа?

Що так сумнівні цифри?

Чому дорівнює гранична відносна похибка добутку?

Чому дорівнює гранична відносна похибка частки?

Назвіть способи розв’язку оберненої задачі теорії похибок.

Як називають комп’ютерну похибку округлення?

Які методи називають чисельними?

Назвіть типи чисельних методів.

Що таке ітерація?

З яких кроків складається алгоритм ітераційного методу?

Що показує збіжність ітераційного методу?

3. ЧИСЕЛЬНЕ РОЗВ'ЯЗУВАННЯ РІВНЯНЬ

Вирази

$$f(x)=0, \tag{3.1}$$

$$x=\varphi(x) \tag{3.1'}$$

називають **рівнянням**. Вираз (3.1) називають рівнянням в **нормальному** вигляді, (3.1') – в **явному**.

Переведення рівняння з **нормального** виду до **явного** проводиться простим множенням обох частин на константу K та додаванням змінної x:

$$K\cdot f(x)+x=x, \quad \varphi(x)=x+K\cdot f(x), \quad f(x)=\varphi(x)-x.$$

Коефіцієнт K завжди можна обрати в такий спосіб, щоб $\varphi(x)$ задовольняла умові

$$0<|\varphi'(x)|=q<1, \tag{3.2}$$

$$\varphi'(x)=1+K\cdot f'(x),$$

$$-q<1+Kf'(x)<q<1.$$

В разі додатних значень похідної $f'(x)$

$$0<min<f'(x)<max, \quad x=(a,b).$$

в якості min та max можна взяти відповідно найменше й найбільше значення $f'(x)$ на відрізку (a,b). Тоді

$$1+K\cdot min < 1+K\cdot f'(x) < 1+K\cdot max.$$

Нерівність (3.2) буде виконано, якщо покласти

$$1+K\cdot max=q, \quad 1+K\cdot min=-q,$$

тобто

$$K=\frac{-2}{min+max}, \quad q=\frac{max-min}{max+min}.$$

Якщо похідна $f'(x)$ не змінює знак на проміжку (a,b), то її значення на кінцях проміжку можна вважати за найменше й найбільше, тобто за min та max.

В такому випадку формула перетворення з нормального вигляду рівняння в явний набуде наступного вигляду

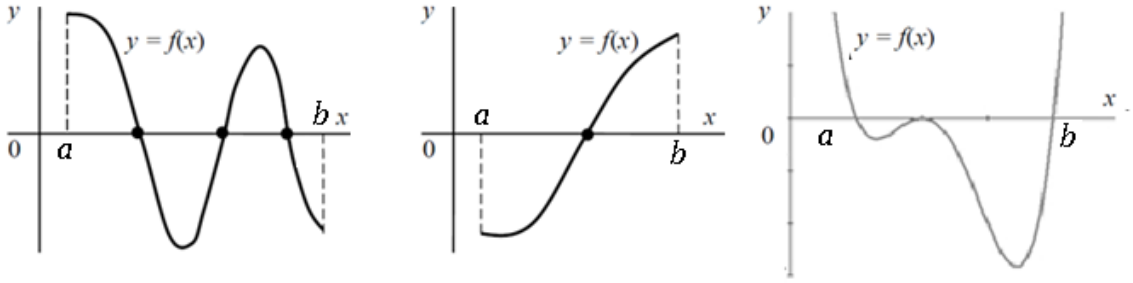
$$\varphi(x) = x - \frac{2}{f'(a) + f'(b)} f(x).$$

Якщо **f(x)** – алгебраїчний багаточлен, то рівняння (3.1) називають **алгебраїчним**, якщо **f(x)** містить якісь математичні функції (lg x, e^x тощо), то рівняння називають **трансцендентним (нелінійним)**.

Множина змінних $x_к$ **(к=1, 2, ...N),** за яких вираз (3.1) стає тотожністю, називаються **нулями** функції **f(x)** чи **коренями рівняння** (3.1).

Рівняння (3.1) може мати один, кілька розв'язків або не мати жодного. Значення аргументу **x**, за якого вираз (3.1) стає тотожністю, але в околі точки x функція **f(x)** не змінює знак називають **парними коренями**.

На графіку функції розв'язок відображується як точка перетинання функції **f(x)** (ординат) та осі **x** (абсцис) (рис. 3.1).



а б в

Рисунок 3.1 – Геометрична інтерпретація розв'язків рівнянь на проміжку (**a,b**): а – рівняння з кількома розв'язками; б – рівняння з одним розв'язком; в – рівняння має парний корінь

Проміжок (**a,b**), на якому є єдиний корінь x* рівняння, називають **проміжком ізоляції**, а сам розв'язок називають **ізольованим** на заданому проміжку.

Методи розв'язання рівнянь діляться на **прямі** та **ітераційні**.

Прямі методи дозволяють знайти розв'язок безпосередньо за допомогою формул відповідного методу і теоретично забезпечують отримання точного значення. **Прямі** методи дають змогу знайти точний

розв’язок системи за допомогою виконання **наперед відомої скінченої** кількості арифметичних операцій у припущенні, що всі обчислення, виконуються точно. На практиці комп'ютерні обчислення виконуються з обмеженою кількістю десяткових розрядів, тому розв’язки, які обчислюються за точними методами, фактично є наближеними числами з певними похибками (округлень, обчислень).

Прямі методи не є універсальними, вони розроблені лише для обмеженого кола рівнянь, наприклад, для алгебраїчних рівнянь степеню $n<5$.

В **ітераційних** методах отриманий розв'язок завжди є наближеним, хоча може бути як завгодно близьким до точного. Вони дають змогу знайти наближений розв’язок системи із заздалегідь указаною точністю шляхом виконання **наперед невідомої скінченої** кількості арифметичних операцій. Точний розв’язок системи за допомогою ітераційних методів можна знайти тільки теоретично як границю збіжного нескінченого процесу.

Можна виділити два типи **ітераційних** методів:

- **Методи звуження інтервалу**, що містить корінь – методи **гарантованої збіжності**. До цього класу відносяться методи половинного ділення, золотого перетину, Фібоначі, послідовного пошуку тощо. Вони є відносно простими, але мають низьку швидкість збіжності.
- **Методи апроксимації**, в яких функція f(x) замінюється деякою більш простою функцією, для якої відшукується корінь – методи **умовної збіжності**. До цього класу відносяться Ньютонівські методи, методи Мюлера, простих ітерацій тощо. Швидкість збіжності у них вище.

У загальному випадку розв'язання рівнянь вирішується в два етапи:

1) **відокремлення кореня**, тобто встановлення інтервалу (a, b), в якому міститься ізольований корінь рівняння, для визначення початкового наближення;

2) **уточнення кореня** до заданого степеня точності за допомогою одного з ітераційних методів.

В якості **критерію** досягнення потрібної точності (граничної абсолютної похибки) в академічних джерелах [2, 7, 9] рекомендується використовувати критерій закінчення ітераційного процесу за аргументом:

$$|x_k - x_{k+1}| < \varepsilon_x.$$

Це означає, що шуканий розв'язок знаходиться всередині діапазону

$$x - \varepsilon_x \le x^* \le x + \varepsilon_x.$$

Для методів звуження інтервалу – $x = \frac{x_k + x_{k+1}}{2}$.

Для апроксимаційних методів – $x = x_{k+1}$.

В комп'ютерній літературі зустрічається критерій останову за значенням функції

$$|f(x_k)| < \varepsilon_f.$$

Вказаний критерій не може вважатися універсальним, бо для функцій, які повільно змінюються, його застосування викликає надзвичайну чутливість від значення початкового наближення, та призводить до випадків, коли знайдене наближення суттєво відрізняється від теоретичного шуканого значення (рис. 3.2).

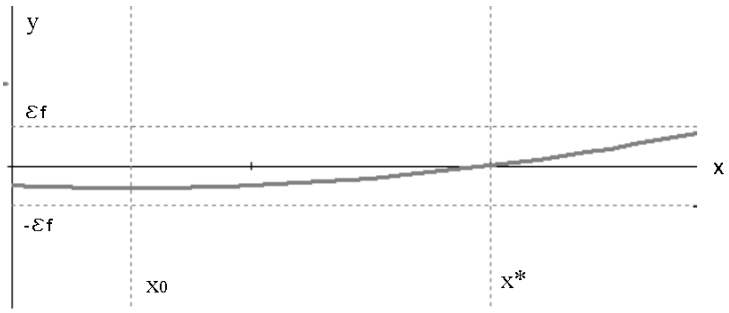


Рисунок 3.2 – Вплив критерію за значенням функції

Критерії за аргументом ε_x та значенням функції ε_f взаємозамінними.

$$|f'(x)| = \left|\frac{df(x)}{dx}\right| \approx \frac{|f(x_{k+1}) - f(x_k)|}{|x_{k+1} - x_k|} = \frac{|f(x_k)|}{\varepsilon_x} = \frac{\varepsilon_f}{\varepsilon_x},$$

$$\varepsilon_x = \min_{x_{k+1}x_k} \frac{\varepsilon_f}{|f'(x)|}.$$

Для знаходження значення похідної можна використовувати наступне твердження [7]: «Якщо в інтервалі (**a,b**) перша та друга похідні функції

мають однакові знаки, то мінімум модуля похідної досягається в точці **a** , якщо знаки першої та другої похідних є різними – в точці **b**».

Відокремлення коренів

Для відокремлення коренів використовується наступна теорема:

Якщо функція **f(x)** неперервна на інтервалі (**a,b**) і якщо на кінцях інтервалу значення функції мають протилежні знаки, то **f(x)** має принаймні один дійсний корінь на інтервалі (**a,b**). Якщо при цьому **f(x)** має першу похідну, що не змінює знак на інтервалі, тобто функція є монотонною, то корінь є єдиним.

Простіше за все визначити всі проміжки ізоляції можна побудувавши графік функції.

Алгоритмічно процес **ізоляції** коренів може мати наступний порядок [22]:

- Розбити первинний інтервал (**a, b**) рівні відрізки з кроком **h** $x_i = a + i \cdot h$.
- Визначити знак функції **f(x)** на кінцях кожного і-го відрізка.
- Перевірити наявність кореню на інтервалі. Якщо знаки однакові та **h** достатньо мале, то можна сподіватися, що на цьому коренів рівняння немає. Якщо знаки різні, то на відрізку існує розв'язок рівняння.
- Перевірити наявність одного кореню оцінюванням знаків похідної **f'(x)** на кінцях відрізка. Якщо крок **h** достатньо малий і знак похідної **f'(x)** на кінцях відрізка не змінюється, то можна сподіватися, що на відрізку корінь один. Таким чином, корінь рівняння можна вважати відокремленим. Якщо знак похідної **f'(x)** змінюється, то впевненості, що корінь один, немає. Відрізки, для яких немає упевненості в тому, що розв'язок рівняння лише один, розбивати вже з меншим кроком, тобто повторити для них описану процедуру відокремлення коренів.

3.1. Методи звуження інтервалу (гарантованої збіжності)

Метод ділення навпіл (бісекції, дихотомії, однієї другої).

Метод застосовується на відомому інтервалі (**a,b**), в якому апріорі знаходиться корінь рівняння. Для використання методу рівняння записується в нормальному виді. Правило методу полягає в діленні інтервалу (**a,b**) навпіл, тобто знаходженні точки с середини інтервалу (рис. 3.3)

$$c = \frac{a+b}{2}. \qquad (3.3)$$

Схема методу

1. Визначення рівняння (3.1), початкових даних – інтервалу (**a,b**), точності **εx** .
2. Знаходження точки середини інтервалу (3.3).
3. Перевірка відповідності довжини інтервалу точності

$$| b-a | < \varepsilon_x.$$

В разі виконання умови середина інтервалу є коренем рівняння. В разі невиконання умови інтервал скорочується вдвічі з боку того краю, на якому знаки функції в точці середини та кінця інтервалу співпадають. Тобто або **a=c**, або **b=c**. Перехід на п.2.

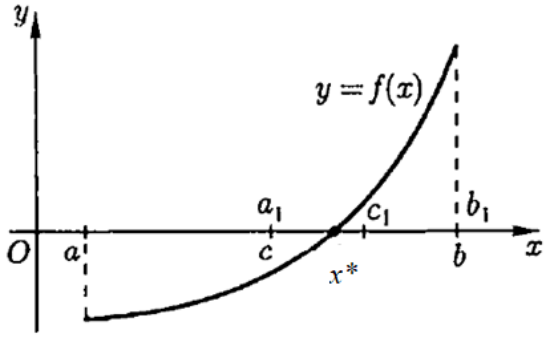


Рисунок 3.3 – Метод дихотомії

Метод «золотого» розтину (метод Фібоначі)

Метод є різновидом методу **ділення навпіл**. Для звуження інтервалу застосовується не просте ділення на 2, а ділення за **правилом «золотого» розтину**, яке збігається з **правилом Фібоначі**.

За правилом «**золотого» розтину** відрізка (**a,b**) довжина всього відрізку так відноситься до довжини більшої частини, як довжина більшої частини відноситься до довжини меншої частини.

$$\frac{|ab|}{|ac|} = \frac{|ac|}{|bc|}$$

$$\frac{|b-a|}{|c-a|} = \frac{|c-a|}{|b-a|-|c-a|} = \frac{1}{k}$$

$$= \frac{k}{1-k},$$

$$k = \frac{\sqrt{5}-1}{2} = 0.61803,$$

c=a+k·(b-a). (3.4)

Схема методу

1. Визначення рівняння (3.1), початкових даних – інтервалу (**a,b**), точності **εx**.
2. Знаходження точки «**золотого**» перетину (3.4).
3. Перевірка відповідності довжини інтервалу точності
   | b-a | < εx.

В разі виконання умови середина інтервалу є коренем рівняння. В разі невиконання умови інтервал скорочується з боку того краю, на якому знаки функції в точці середини та кінця інтервалу співпадають. Тобто або **a=c**, або **b=c**. Перехід на п.2.

Метод послідовного пошуку (метод послідовного наближення, метод ітерацій)

Метод застосовується на відомому інтервалі (**a,b**), в якому апріорі знаходиться корінь рівняння. Для використання методу рівняння записується в нормальному вигляді (3.1).

Схема методу

1. Визначення рівняння (3.1), початкових даних – інтервалу (**a,b**), точності **εx,** кількості підінтервалів **N** .
2. Знаходження кроку пошуку **h=(b-a)/N**.
3. Знаходження точок першого підінтервалу **a1=a, b1=a+h**.
4. Перевірка знаходження кореня в підінтервалі шляхом перевірки знаків функції на кінцях підінтервалу. В разі, коли кореня всередині немає, підінтервал зсувається на один крок **a1=a1+h, b1=b1+h**.

Перехід на п.4. В разі, коли корінь всередині є проводиться перевірка відповідності довжини підінтервалу точності **|b1-a1|<εₓ.** В разі виконання умови середина інтервалу є коренем рівняння.

5. В разі невиконання умови проводиться зміна напрямку пошуку на протилежну та зменшення кроку пошуку **h=-h/N, a1=b1, b1=a1+h**. Перехід на п.4.

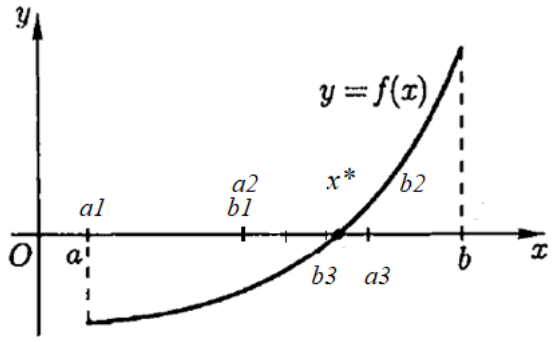


Рисунок 3.4 – Метод послідовного пошуку

Метод хорд

Метод хорд є методом гарантованої збіжності на інтервалі (**a,b**), хоча іноді його відносять до **Ньютонівських** методів Умовою застосування методу є зміна знаку монотонної функції **f(x)** на кінцях інтервалу.

В даному методі функція **f(x)** замінюється прямою – хордою, яка проходить через точки кінців інтервалу пошуку. В якості наближення ітерації приймається точка перетину хорди з віссю OX.

Рівняння хорди має наступний вигляд:

$$\frac{y-f(a)}{f(b)-f(a)}=\frac{x-a}{b-a}.$$

Для точка $\mathbf{x_1}$ перетину хорди з віссю OX (**y=0**) можна записати:

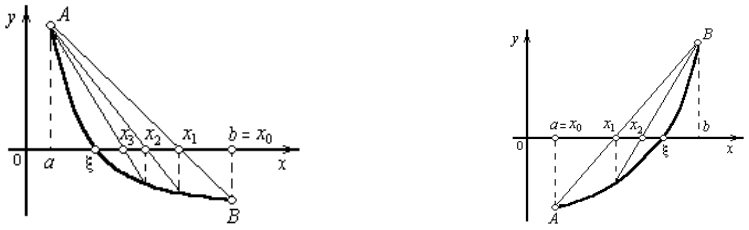
$$x_1 = a - \frac{f(a)}{f(b)-f(a)}(b-a). \qquad (3.5)$$

Отриманий вираз (3.5) є ітераційним наближуючим правилом методу хорд.

Подальші наближення залежать від властивостей функції **f(x).** Для опуклих функцій з **f″(x) > 0** на інтервалі пошуку **(a,b)** функція

y=f(x) розташована нижче хорди. Для спадаючих функцій з **f'x) < 0** значення похідної в точці **a** **f(a) > 0** (рис. 3.6, а) для зростаючих функцій з **f'(x) > 0** значення похідної в точці **b** **- f(b) > 0** (рис. 3.6, б).

Випадок увігнутих функцій **f″(x) < 0** змінюється записом рівняння зі знаком «мінус» **- f(x) = 0**.



а б

Рисунок 3.6 – Метод хорд для опуклих функцій: а – зростаюча функція; б – спадаюча функція

В першому випадку точка **a** вважається нерухомою та інтервал пошуку звужується з боку точки **b**:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k)-f(a)}(x_k - a). \qquad (3.6)$$

В другому випадку точка **b** вважається нерухомою та інтервал пошуку звужується з боку точки **a**:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(b)-f(x_k)}(b - x_k). \qquad (3.7)$$

В якості нерухомого визначається той кінець інтервалу пошуку, для якого знак функції **f(x)** співпадає зі знаком її другої похідної **f″(x)** .

Особливістю методу є те, що ітераційні наближення $\mathbf{x_к}$ складають монотонну послідовність та лежать по той бік кореня **x*,** де функція **f (x)** має знак, протилежний знаку її другої похідної **f″(x).**

Схема методу

1. Визначення рівняння (3.1), початкових даних – інтервалу (**a,b**), точності **εx**
2. Визначення нерухомої точки.

3. В разі нерухомої точки **a** знаходження наближення ітерацій за правилом (3.6), разі нерухомої точки **b** знаходження наближення ітерацій за правилом (3.7),
4. Перевірка досягнення точності **εx**. В разі досягнення точності **x2** є коренем. В разі не досягнення точності – зміна рухомої точки та повторення п.4.

3.2. Методи умовної збіжності

Безперечною достоїнством методів умовної збіжності є те, для уточнення значення кореня рівняння на відрізку **(a,b)** достатньо визначити тільки **одну** точку початкового значення.

Обмеженістю методів **умовної** збіжності є необхідність попереднього аналізу поведінки функції рівняння з метою її відповідності до умов застосування того чи іншого методу умовної збіжності.

Метод простої ітерації

Для застосування методу простої ітерації рівняння повинно бути записано в явному вигляді (3.1') .

Графічно розв'язання рівняння виглядає не як знаходження точки перетину функції **f(x)** осі абсцис OX, а як знаходження точки перетину прямих **y=x** та **y=ϕ(x)**

$$\begin{cases} y(x) = x \\ y(x) = \varphi(x) \end{cases}.$$

Умовою збіжності методу простої ітерації є вираз (3.2)

$$|\varphi'(x)| < 1.$$

Дійсний корінь **x*** рівняння є абсцисою точки M перетину кривої **y=φ(x)** та прямої **y = x** .

Ітераційний процес будує ламану у вигляді «сходин» або спіралі в залежності від властивостей функції рівняння.

Ламана $x_0A_0B_1A_1B_2A_2\ldots$ («сходи», рис. 3.7. а) виходить, якщо похідна **φ′(x)** є додатною. Ланки ламаної почергово є паралельними осі **Ox** та осі **Oy**, вершини **Ai** лежать на кривій **y=φ(x),** а вершини **Bi** – на прямій **y=x**.

Спільні абсциси точок **Ai** та **Bi** є послідовними наближеннями **xi** до кореня **x*.**

Ламана $x_0A_0B_1A_1B_2A_2$ **...** – є спіраллю (рис. 3.7 б) виходить, якщо похідна $\varphi'(x)$ є від'ємною.

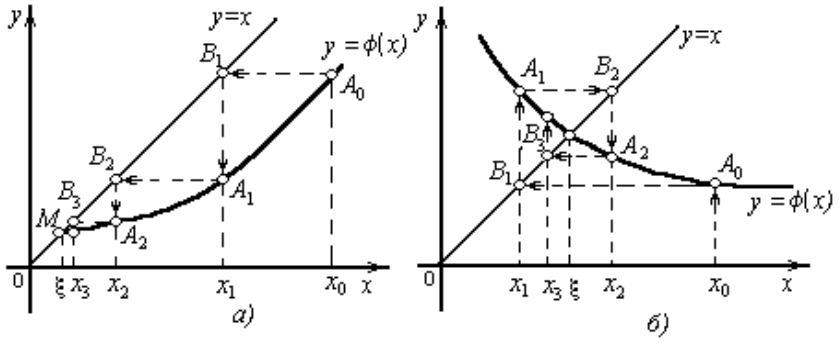


Рисунок 3.7 – Метод простої ітерації

В разі невиконання умови застосування методу, тобто коли **|φ'(x)|>1**, ітераційний процес буде розходитися (рис. 3.8).

Метод простої ітерації має лінійну збіжність.

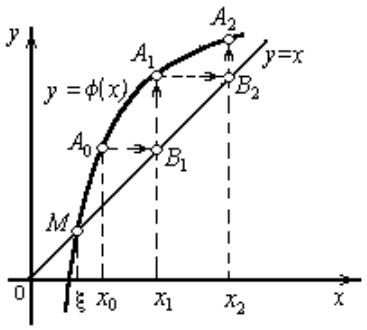


Рисунок 3.8 – Невиконання умови (3.2)

Схема методу

1. Визначення початкових значень: **x0, εx, ϕ(x).**
2. Визначення значення першого кроку ітерації **x1 =** φ**(x0).**
3. Перевірка досягнення точності ε_x**.** В разі досягнення точності **x1** є коренем. В разі не досягнення точності зміна початкової точки на точку наближення **x0=x1** та повторення п.2.

Методи Ньютона

Найбільш розповсюдженими серед методів розв'язання рівнянь є методи Ньютона.

Методи застосовуються для рівнянь, що записані в **нормальному** вигляді (3.1).

Застосування Ньютонівських методів потребує виконання наступних умов:

- функція **f(x)** має похідну **f'(x)** на інтервалі (**a,b**);
- функція **f(x)** не має екстремумів на інтервалі (**a,b**). Тобто перша похідна та друга похідні не змінюють знаків на інтервалі (**a,b**).

В Ньютонівських методах реальна функція замінюється лінійною функцією (прямою), тому методи Ньютона називають методами **лінеаризації** або **лінеаризованими** методами.

Метод Ньютона (дотичних)

Розкладання функції **f(x)** в ряд Тейлора в околі точки x_0 в разі, коли вона є близькою до точки **x*** можна записати як

$$f(x*) = f(x_0) + (x* - x_0)f'(x_0) + \frac{f''(x_0)}{2!}(x* - x_0)^2 + \cdots$$

Для лінійних членів ряду вираз можна записати у вигляді

$$f(x*) = f(x_0) + (x* - x_0)f'(x_0).$$

Оскільки x* є коренем рівняння, то **f(x*)=0** та

$$x* = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

В разі, коли **f(x)** є квадратичною функцією, то корінь буде знайдено за один крок. Для інших функцій потребується кілька кроків ітерації для наближення до теоретичного значення кореня із визначеною точністю **εx**. При цьому **k+1**-те наближення знаходиться через попереднє за виразом (3.8)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \qquad (3.8)$$

Метод **Ньютона** називають **методом дотичної**, бо взявши з ряду тільки перші два доданки, функцію замінено на пряму, яка є дотичною до функції в точці **x0**.

В більшості випадків для уточнення значення кореню достатньо в якості початкового наближення визначити точку в околі **x*.** В деяких джерелах [20, 22] рекомендується «в якості вихідної точки **x0** обирати той кінець інтервалу (**a,b**), якому відповідає ордината того ж знаку, що й знак другої похідної **f″(x)** в тій же точці».

Геометрична інтерпретація методу дотичних зображена на рис. 3.9.

В точці початкового значення аргументу **x0** будується перпендикуляр до осі OX. В точці його перетину з функцією **f(x)** проводиться дотична. Реальна крива замінюється побудованою дотичною та знаходиться наближення кореню як точка перетину **x1** осі абсцис OX з дотичною.

Рівняння дотичної, яка проходить через точку **x0** має вигляд

y—f(x0)=f'(x0)(x—x0).

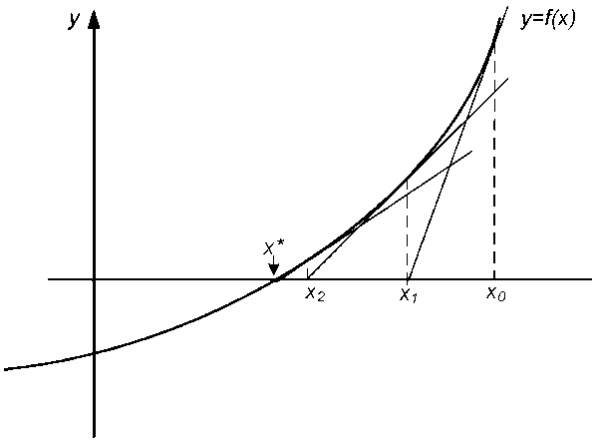


Рисунок 3.9 – Метод дотичних

Для знаходження точки перетину цієї прямої з віссю OX потрібно підставити в рівняння дотичної **x=x1, y=0**.

—f(x0)=f'(x0)(x1—x0), звідки **x1=x0—f(x0)/f'(x0)** .

Наступна ітерація проводиться з точки **x1** для знаходження наближення **x2**. Процес повторюється доки точка **xк** не наблизиться до шуканого значення кореню із визначеною точністю ε_x.

Схема методу

1. Визначення початкових даних: **x0, εx, f(x), f'(x).**
2. Знаходження наближення першого кроку ітерацій за правилом (3.8).
3. Перевірка досягнення точності **εx.** В разі досягнення точності **x1** є коренем. В разі не досягнення точності зміна початкової точки на точку наближення **x0=x1** та повторення п.2.

Метод дотичних має квадратичну швидкість збіжності.

Недоліками метода Ньютона є необхідність аналітичного визначення похідної функції **f'(x)** та необхідність на кожному кроці крім значення самої функції **f(x),** розраховувати значення її похідної **f'(x)** .

Модифікований метод Ньютона (метод паралельних дотичних, partan метод)

Метод був розроблений для зменшення розрахунків шляхом прибирання обчислення значення похідної **f'(x)** на кожному кроці ітерацій.

Для гладких функцій, які незначно змінюють крутизну, можна припустити сталість значення похідної в інтервалі пошуку

$$f'(x)_{(a\ b)}=const=f'(x0).$$

В такому разі **k+1**-те наближення знаходиться через попереднє за виразом

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}. \qquad (3.8')$$

Схема методу не відрізняється від схеми методу дотичних з урахуванням заміни правила ітерацій (3.8) на (3.8').

Геометрична інтерпретація методу паралельних дотичних зображена на рис. 3.10.

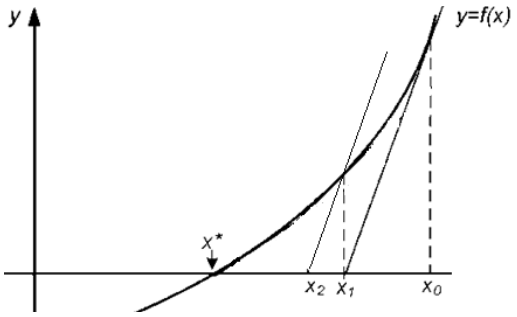


Рисунок 3.10 – Метод паралельних дотичних

Метод січних (secant метод)

Метод розроблено для того, щоб в методі дотичних не визначати аналітично похідну функції **f'(x).**

Геометрична інтерпретація методу січних зображена на рис. 3.11.

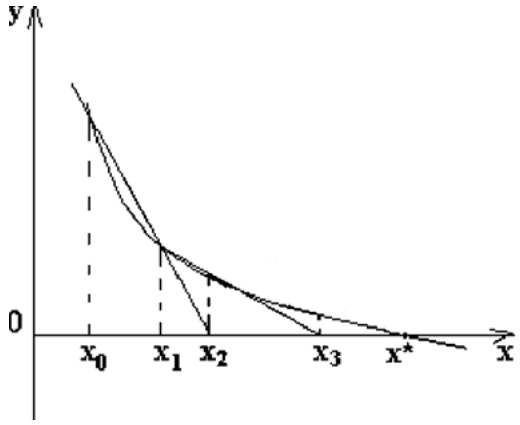


Рисунок 3.11 – Метод січних

Особливістю метода є те, що на кожному кроці використовуються **три** точки x_{k-1}, x_k, x_{k+1}, тобто метод є **трьохточковим**.

Поряд з точкою початкового значення **x0** визначаємо точку **x1** Через точки **f(x0)** и **f(x1)** проведемо пряму – січну. Замінимо функцію **f(x)** побудованою січною. Точку наближення ітерації **x2** визначимо як точку перетину січної з віссю OX.

Рівняння прямої, яка проходить через точки **x0, x1**:

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - f(x_0)}{f(x_1) - f(x_0)}.$$

Для точки наближення x_2 значення функції дорівнює нулю **y=0**.

$$x_2 = x_0 - \frac{f(x_0)}{f(x_1) - f(x_0)}(x_1 - x_0).$$

В разі невиконання умов точності точки **x0, x1** пересуваються **x0=x1, x1=x2,** будується нова січна та знаходиться наближення як точка перетину січної з віссю OX.

Відповідно до визначення похідної

$$f'(x) = \lim_{dx \to 0} \frac{f(x) - f(x - dx)}{dx}.$$

За умови $\mathbf{x=x_к}$, $\mathbf{x-dx=x_{k-1}}$,

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Тоді ітераційне правило набуде наступного вигляду

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}). \quad (3.9)$$

Для всіх кроків, починаючи з другого, точки відомі. Для першого кроку точка $\mathbf{x_{-1}}$ невідома. В якості неї можна взяти точку, яка знаходиться в околі точності розв'язання $\boldsymbol{\varepsilon x}$ від точки початкового значення $\mathbf{x1=x0+C\cdot\varepsilon x}$, де $\mathbf{C=1…3}$.

Схема методу

1. Визначення початкових даних: $\mathbf{x0,\varepsilon x,f(x),f'(x),C,}$ $\mathbf{x1=x0+C\cdot\varepsilon x}$.
2. Знаходження наближення першого кроку ітерацій за правилом (3.9).
3. Зміна початкових точок $\mathbf{x0=x1,\ x1=x2}$.
4. Перевірка досягнення точності $\boldsymbol{\varepsilon x}$. В разі досягнення точності $\mathbf{x2}$ є коренем. В разі не досягнення точності повторення п.3.

3.3. Засоби розв'язання рівнянь СКМ MathCAD, Matlab

Засоби MathCAD

В СКМ MathCAD для **чисельного** розв'язання рівнянь призначені функції **root, polyroots, given/find.** Функція **root** розв'язує нелінійні рівняння, функція **polyroots** – алгебраїчні рівняння. Блок **given** безпосередньо призначений для розв’язання систем нелінійних рівнянь, але його теж можна використовувати для знаходження коренів рівнянь.

Для розв'язання алгебраїчних рівнянь функція **polyroots** може використовувати метод **Лагера** або метод **супроводжуючої матриці,** який дозволяє отримати трохи більш точний розв'язок. За замовчанням встановлено більш стабільний метод **Лагера.** Контекстне меню функції (рис. 3.12) дозволяє змінити метод розв'язання.

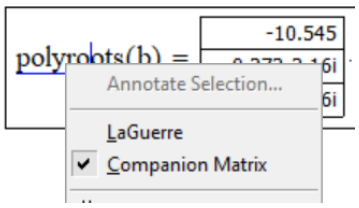


Рисунок 3.12 – Контекстне меню функції **polyroots**

Точність розв'язку рівняння функцією **polyroots** не залежить від значення змінної **TOL**. Функція знаходить всі корені алгебраїчного рівняння та повертає результат у вигляді вектора-стовпця.

Аргументом функції **polyroots** є вектор-стовпець коефіцієнтів алгебраїчного рівняння. Коефіцієнти мають розташовуватися в порядку збільшення степеню змінної рівняння. Тобто починатися з елемента, що має нижчий степінь.

Наприклад, для алгебраїчного рівняння

$$2x^3 + 20x^2 - 2x + 100 = 0 \quad . \tag{3.10}$$

застосування функції **polyroots** може мати наступний вигляд:

$$\text{polyroots}\begin{pmatrix}100\\-2\\20\\2\end{pmatrix} = \begin{pmatrix}-10.545\\0.272-2.16i\\0.272+2.16i\end{pmatrix} \qquad b := \begin{pmatrix}100\\-2\\20\\2\end{pmatrix} \qquad \text{polyroots}(b) = \begin{pmatrix}-10.545\\0.272-2.16i\\0.272+2.16i\end{pmatrix}$$

Функція **root** слугує для знаходження кореня рівняння чисельними методами. Обмежень на вид рівняння немає. В функції реалізовано два методи: різновид метода **січних** та різновид метода **дихотомії.**

Метод січних за алгоритмом **Мюлера** для заходження чергового наближення кореня використовує три попередні точки, тобто квадратичну інтерполяцію функції рівняння.

Метод дихотомії за алгоритмом **Брента** комбінує вибір точок діленням відрізку навпіл та метод **Рідера** з квадратичною зворотною інтерполяцією та пошуком кореня **x** через квадратичні функції експонент **f(x).**

Для застосування методу **січних** в функції **root(f(x),x)** використовуються два аргументи: вираз рівняння у вигляді імені функції

користувача чи безпосередньо виразу функції рівняння в нормальному вигляді та змінна, відносно якої необхідно розв'язання рівняння. Змінна повинна мати попередньо визначене чисельне значення початку пошуку.

Наприклад,

$z := 5 \quad y(x) := \sqrt[3]{x - 5}$

$\text{root}\left(\sqrt[3]{z - 5}, z\right) = 5$

$\text{root}(y(z), z) = 5$

Для використання методу **дихотомії** в функції **`root(f(x),x,a,b)`** після імені змінної рівняння треба додати два параметри: початкову **`a`** та кінцеву **`b`** точки діапазону пошуку кореня у вигляді чисельних значень або попередньо визначених змінних.

Наприклад,

$z := 5 \quad y(x) := \sqrt[3]{x - 5} \quad a := 1 \quad b := 7$

$\text{root}\left(\sqrt[3]{z - 5}, z, 1, 7\right) = 5$

$\text{root}(y(z), z, a, b) = 5$

Функція може бути застосована для розв'язання рівнянь з параметрами. Для таких рівнянь значення параметра повинно бути попередньо визначено явно у вигляді чисельного значення.

Наприклад, параметр можна визначити поза функцією рівняння:

$a := 2 \quad y(x) := a \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100 \quad z := 5$

$\text{root}(y(z), z) = -10.545$

.

Параметр рівняння також може бути введено аргументом функції рівняння:

$ya(a, x) := a \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100$

$\text{root}(ya(1, z), z) = -20.34$

.

Другий спосіб є більш правильним.

Функція **root** може бути застосована всередині функції користувача.

Наприклад:

$y(a,x) := x - a$

$b(f,p,z) := root(f(p,z),z) \qquad z := 4 \qquad b(y,0.6,z) = 0.6$

В документації СКМ стверджується, що чисельний корінь рівняння знаходиться функцією **root** з точністю, значення якої визначається системною змінною **TOL**.

Слід відмітити, що до MathCAD 13 критерієм розв'язання рівняння функцією **root** слугувала умова того, що абсолютне значення функції рівняння в точці наближення $\mathbf{y(x_i)}$ є меншим за значення абсолютної похибки **TOL:**

$$|y(x)| \leq TOL.$$

Такий критерій зустрічається в деяких джерелах комп'ютерної тематики.

Це призводило до того, що функція **root** визначала різні корені для тотожних з математичної точки зору рівнянь, що суперечило математичним законам. Наприклад, таким рівняннями є рівняння **y(x)=0** та це рівняння, помножене на сталу **k≤1: k·y(x)=0.**

В версіях з MathCAD 14 критерій було змінено на більш математично вірний

$$|x_{i+1} - x_i| \leq TOL.$$

Наприклад, для рівняння (3.10) та тотожно зміненого рівняння

$$0.001 \cdot (2x^3 + 20x^2 - 2x + 100) = 0 \qquad (3.10')$$

застосування функції **root** дає однакове значення кореня:

$y(x) := 2 \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100 \qquad z := -5$

$s(x) := 10^{-3} \cdot y(x)$

$root(s(z),z) = -10.545 \qquad root(y(z),z) = -10.545$

Застосування функції **root** потребує обов'язкового аналізу отриманих результатів.

Використання функції **root** в режимі метода **січних** може привести до неочікуваного або невірного результату.

Наприклад, для рівняння (3.10) графік (рис. 3.13) показує наявність одного дійсного кореня в околі значення **x= -10.5**.

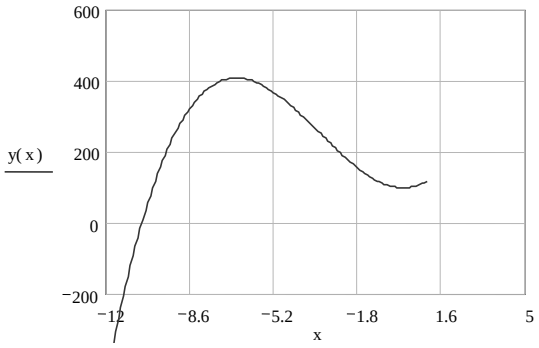


Рисунок 3.13 – Графік функції (3.10)

Застосування функції з початковим наближенням **x=-5** видає вірне дійсне значення кореня **-10,545**. Визначення початкового наближення **x=0** призводить комплексного значення, яке є коренем рівняння, але не є дійсним числом.

$$y(x) := 2\cdot x^3 + 20\cdot x^2 - 2\cdot x + 100 \quad z := -0$$

$$root(y(z), z) = 0.272 + 2.16i$$

Коренями рівняння $\mathbf{f(x)=x^2+0.0001=0}$ є комплексні числа **x=±0.01i**. Використання метода січних в функції **root** з точністю за замовчанням **(TOL=0.001**) дає дійсний корінь **x=0.018**. Формальний критерій співвідношення між значенням функції та похибкою виконано: $\mathbf{f(0.018)=4.225*10^{-4}<0.001}$. Проте з позицій теоретичної математики знайдене значення не можна суворо вважати коренем рівняння:

$$y2(x) := x^2 + 0.0001$$

$$x2 := root(y2(z), z) \quad x2 = 0.018$$

$$y2(x2) = 4.225 \times 10^{-4}$$

.

Для знаходження вірного значення кореню можна спробувати зменшити похибку нижче за значення вільного члена рівняння:

$y2(x) := x^2 + 0.0001 \quad TOL := 10^{-4}$

$x2 := root(y2(z), z) \quad x2 = -0.01i$

$y2(x2) = 0$

Слід зазначити, що спосіб зменшення значення похибки не є універсальним. Наприклад, функція **root** знаходить для рівняння $\frac{1}{x} = 0$ корінь x, який формально задовольняє критерію закінчення пошуку для будь-якого малого значення похибки:

$y2(x) := \frac{1}{x}$

$x2 := root(y2(z), z) \quad x2 = 3.297 \times 10^{10}$

$y2(x2) = 3.033 \times 10^{-11}$

.

Проте рівняння з точки зору математики є некоректним та не має взагалі коренів.

Значення початкового наближення має особливу вагу для трансцендентних рівнянь, які мають декілька коренів. Функції **root** з початковою точкою пошуку зі значенням похідної, що наближене до нуля, може видати результат, який буде значно віддаленим від очікуваного значення. Чим ближче до нуля є значення похідної функції рівняння в початковій точці, тим більшим буде значення знайденого кореню.

Наприклад, рівняння **sin(x)=0** має корені **x=пк**, де к – ціле число. Звичайний виклик функції **root** дозволяє отримати за один виклик тільки один корінь

$z := 1 \qquad root(\sin(z), z) = -5.126 \times 10^{-13}$

.

Проаналізуємо залежність винайдених функцією **root** коренів від початкового значення. Для цього визначимо початкове значення ранжованою змінною в околі ±0.1 від точки **x=п/2** з кроком 0.02 та застосуємо функцію **root** для визначеної ранжованої змінної всіх початкових значень.

$x := \frac{\pi}{2} - 0.1, \frac{\pi}{2} - 0.08 .. \frac{\pi}{2} + 0.1$ $y(z) := root(sin(z), z)$

x =	$\frac{y(x)}{\pi}$ =	sin(y(x)) =
1.4708	-2.0000	$-2.107 \cdot 10^{-14}$
1.4908	-2.0000	$1.357 \cdot 10^{-9}$
1.5108	-4.0000	$2.137 \cdot 10^{-13}$
1.5308	9.0000	$1.102 \cdot 10^{-15}$
1.5508	$9.1040 \cdot 10^{3}$	$-1.74 \cdot 10^{-12}$
1.5708	$-7.2900 \cdot 10^{4}$	$1.329 \cdot 10^{-11}$
1.5908	12.0000	$5.636 \cdot 10^{-15}$
1.6108	74.0000	$-1.957 \cdot 10^{-15}$
1.6308	58.0000	$-8.811 \cdot 10^{-13}$
1.6508	3.0000	0
1.6708	4.0000	0

Аналіз наведених результатів показує, що для всіх початкових точок в визначеному околі функція знайшла вірні корені. Абсолютна похибка для всіх точок не перевищує 10^{-9}. Але замість очікуваних значень кореня **x=π** знайдені значення кореню в обраному діапазоні початкових точок віддалені від очікуваного значення від 2-х до 10000 періодів.

Слід зазначити, що розрахунки в діапазоні за умови визначення початкового наближення за допомогою ранжованих змінних разом в функції **root** в «класичних» версіях MathCAD потребує обережності.

Наприклад, проведемо аналіз залежності знайденого кореня функцією **root** тотожних математично рівнянь (3.10) та (3.10') від положення початкової точки пошуку в діапазоні від -1 до 0.2 з кроком 0.1:

start := −1 stop := 0.2 dx := 0.1

zz := start, start + dx .. stop

zz =	root(y(zz), zz)	root(s(zz), zz)
-1	-10.545	-10.545
-0.9	-10.545	-10.545
-0.8	-10.545	-10.545
-0.7	-10.545	-10.545
-0.6	-10.545	-10.545
-0.5	-10.545	-10.545
-0.4	-10.545	-10.545
-0.3	-10.545	-10.545
-0.2	-10.545	-10.545
-0.1	-10.545	-10.545
0	-10.545	0
0.1	0.272-2.16i	0.1
0.2	-10.545	-10.545

z0 := 0 z01 := 0.1

root(s(z01), z01) = 0.1

root(y(z01), z01) = 0.272 − 2.16i

root(s(z0), z0) = 0.272 + 2.16i

root(y(z0), z0) = 0.272 + 2.16i

Аналіз результатів показує, що для початкової точки **x=0.1** функція **root** для рівняння **y(x)** (3.10) знаходить замість дійсного кореня **x=-10.545** комплексний корінь **x=0.272-2.26i**. Розрахунок для одиночної початкової точки показує, що функція знаходить комплексний корінь **x=0.272+2.26i** ще й для початковій точці **x=0**. Обидві відповіді є коренями рівняння.

Для модифікованого рівняння **s(x)** (3.10') функція **root** змінює знайдені корені для значень ранжованої змінної початкової точки 0 на **x=0** та для точки 0.1 на **x=0.1**. Обидві відповіді є невірними. Розрахунок для одиночної початкової точки показує, що для точки 0 відповіддю є комплексне значення **x=0.272+2.26i**, що є коренем рівняння. Для початкової точки 0.1 функція **root** визначає невірний результат незалежно від типу початкової точки.

З вищенаведеного можна зробити висновок, що до результатів комп'ютерних обчислень слід відноситися критично. Знайдені функцією **root** корені потребують перевірки на достовірність. Перед застосуванням функції **root** слід проводити аналіз поведінки функції рівняння графічно або аналітично, та аргументовано обирати початкову точку пошуку.

З урахуванням того, що для алгебраїчного рівняння функція рівняння може бути записана у вигляді добутку

$$f(x) = (x - x_1)(x - x_1) \dots (x - x_N),$$

де **xi** – корені рівняння, **N** – степінь полінома, послідовне застосування функції **root** дозволяє знайти всі корені рівняння.

Перший корінь **xi** шукається для вихідного рівняння **h(x)=f(x)** степеню **N**, подальші – для функції степеню **N-1,** яка є часткою ділення вихідної функції на різницю **x-xi**:

$$h(x) = \frac{h(x)}{x - x_i},$$

де **xi** – корінь рівняння, який знайдено на попередньому кроці.

Для розв'язання рівнянь методом січних можна надати наступні рекомендації [12]:

- точку початкового наближення слід обирати якомога ближче до очікуваного значення кореня таким чином, щоб сікуча з точки початкового наближення була направлена в бік кореня;
- не слід обирати початкову точку в околі екстремумів функції рівняння, де похідна має абсолютне значення менше за 2;
- в разі, коли зміна початкової точки не призводить до знаходження кореня, можна спробувати замінити вихідну функцію **f(x)** часткою **g(x)** від ділення вихідної функції на її похідну:

$$g(x) = \frac{f(x)}{f'(x)}$$

.

Функція **g(x)** має нулі в тих же точках, що й вихідна **f(x)**, але має зворотну поведінку: в зонах, де функція **f(x)** змінюється швидко, **g(x)** змінюється повільно, в зонах, де функція **f(x)** змінюється повільно, **g(x)** змінюється швидко.

Засоби Matlab

В пакеті **MATLAB** для чисельного знаходження коренів передбачені вбудовані функції розв'язання нелінійних та для алгебраїчних рівнянь.

Функція **fzero** призначена для чисельного розв'язання трансцендентних рівнянь. Функція використовує метод **Декера**, який є комбінацією **методів дихотомії**, **січних та зворотної інтерполяції**.

Функція
[x,fval,exitflag,output]=fzero(fun,x0,<[x1x2],opt,...
P1,P2,...>) шукає чисельне ДІЙСНЕ значення **x**, яке є нулем функції **fun**.

Обов'язковими параметрами є **fun** – функція рівняння та **x0** – початкова точка початку пошуку у вигляді числам або імені наперед визначеної змінної.

Функція рівняння **fun** може задаватися кількома способами:

1) Покажчиком на в m-файл функцію користувача, в якому описано рівняння

```
function ff = test(v)        x= fzero (@test,0,1)
ff = sin(pi*v).
```

2) Рядком з іменем функції m-файлу функції користувача

```
function ff = test(v)        x=fzero('test',0,1)
ff = sin(pi*v).              x=fzero ('test(x)',0,1)
```

3) За допомогою «**inline**» функції

```
f=inline('3*cos(pi*x)');   x=fzero(f,1)
```

4) Покажчиком на анонімну функцію

```
ms=@(x)3*cos(x)              x=fzero(@ms,1)
```

5) Анонімною функцією

```
                             x=fzero(@(x)sin(x*x),
                             1);
```

Формула, яка обрана в апострофи для виклику за варіантом 2, може використовувати в якості аргументу тільки змінну **x**. Використання інших імен не дозволяється.

Розробник рекомендує використовувати виклики за покажчиком та іменем функції (варіанти 1 та 2). Особливістю таких записів є те, що обробляється тільки один аргумент. В разі параметризації функції кількома аргументами рекомендується застосування вкладених чи анонімних функцій.

Процес пошуку кореня функцією **fzero** складається з двох етапів. На першому етапі в околі точки **x0** визначається діапазон, в якому знаходиться корінь. На другому етапі проводиться уточнення значення кореню всередині цього діапазону. Опціональний вектор інтервалу пошуку кореня **[x1 x2]** може вводитися замість значення **x0**. Його визначення скорочує час пошуку кореня за рахунок першого етапу алгоритму.

Опціональна структура **opt** застосовується в разі потреби визначити особливі умови застосування функції **fzero**. Структура є спільною для багатьох додатків Matlab та містить біля 50-ти полів.

Керування структурою **opt** здійснюється командою **optimset**.

Аргументи структури **optimset** задаються попарно:

options = optimset(...,назва поля, значення,...).

Функція **fzero** використовує поля структури з такими назвами:

Display – керування виведенням інформації про хід пошуку кореня на екран: **'off'** – без виведення / **'iter'–** виведення інформації по кожній ітерації / **'final'–** виведення результуючою інформації / **'notify'** (default) – виведення інформації в разі виникнення проблем.

FunValCheck – керування перевіркою рівняння на комплексні та невизначені значення: **'on'** / **'off'** (default).

OutputFcn – посилання на функцію користувача, яка буде виконуватися на кожній ітерації пошуку кореня.

PlotFcns – масив комірок посилань (cell array) які виводять на екран зображення процесу пошуку кореня. За замовчанням – порожній масив []. Припустимі наступні значення елементів масиву: @**optimplotx** – графік точок, @**optimplotfval** – графік значень функції рівняння на етапі уточнення кореня.

TolX – визначення точності по **x**. За замовчанням розв'язання проводиться з точністю 10^{-6}.

P1,P2,… – опціональні додаткові параметри (до десяти), які можливо передати в функцію рівняння **fun(x,P1,P2,…).**

fval – опціональне значення функції рівняння в точці **x**.

Функції **fzero** крім знайденого значення кореня **x** може повертати додаткову інформацію про умови проведення розв'язання:

exitflag – чисельний код результату дій: 1 – знайдено корінь, -1 – алгоритм перервано, -3 – під час дії отримано невизначене (0/0) чи нескінченне значення, -4 – виникло комплексне значення при діях всередині інтервалу **[x1 x2]** , -5 – точка розриву, -6 – не виявлено зміни знаку функції.

output – структура з додатковою інформацією в наступних полях:

algorithm – **'bisection'**,**'interpolation'**,

funcCount – кількість обрахувань функції,

intervaliterations – кількість ітерацій для знаходження інтервалу пошуку,

iterations – кількість ітерацій пошуку кореня.

Наприклад, звернення до функції

```
>> [x f exi opt]=fzero(inline('x^3-1'),2)
```

повертає корінь **x**, значення функції **f**, код результату завершення дії **exi**, структуру пояснень **opt** з інформацією про кількість кроків ітерацій, кількість обчислень функції, застосований алгоритм:

```
x =      1
f =      0
exi =      1
opt =
    intervaliterations: 10
            iterations: 8
             funcCount: 28
             algorithm: 'bisection, interpolation'
 message: 'Zero found in the interval [0.72, 2.9051]'
```

Для розв'язання рівняння з **N** параметрами функція користувача **fun** повинна бути описана як залежність від **N+1** змінної: **N** параметрів та основна змінна.

Наприклад, розв'язання параметричного рівняння **y=a/((x-0.3)^2+0.01)-6** з параметром **a** відносно змінної **x** можна провести наступним чином:

1) Рядком з іменем функції m-файлу функції користувача

```
function [ y ] = mzero(a, x )
y = a/((x - 0.3)^2 + 0.01)-6;
```

```
fzero('mzero(1,x)',1)
```

2) Анонімною функцією та функцією користувача

```
function [ y ] = mzero(a, x )
y = a/((x - 0.3)^2 + 0.01)-6;
```

```
a=1;
fzero(@(x)
mzero(a,x,1)
```

3) Вкладеною функцією

```
function y = myzero(x,a,x0)      x=myzero(x,1,1)
y = fzero(@(x) mzro(a,x),x0);
 function y = mzro(x,a)
 y = a/((x - 0.3)^2 + 0.01)-6;
 end % end mzro
end  % end myzero
```

Слід зазначити, що функція **fzero** надає більше можливостей для проведення досліджень, її результат є суттєво інформативнішим за засоби розв'язання рівнянь СКМ MathCAD. Проте її застосування потребує попереднього аналізу поведінки функції рівняння. Наприклад, рівняння, яке має комплексні корені в СКМ MathCAD розв'язується простим викликом функції **root**. Виклик функції **fzero** для такого рівняння скоріш за все видасть повідомлення про помилку.

```
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -1.7162e+154 is Inf.)
Check function or try again with a different starting value.
```

Комплексний корінь функцією **fzero** може бути знайдений тільки викликом функції з початковою точкою у вигляді вектора діапазону пошуку [**a b**] з комплексними значеннями такими, щоб функція **fun** мала на кінцях діапазону дійсне значення протилежного знаку.

Функція **roots** призначена для чисельного розв'язання алгебраїчних рівнянь. Вона використовує метод **власних значень супроводжуючої матриці**.

Функція **roots(a)** має один обов'язковий аргумент **a** – вектор-рядок коефіцієнтів алгебраїчного рівняння. Коефіцієнти мають розташовуватися з більшого степеня змінної до меншої. Наприклад, для рівняння $3x^2+2x-2=0$ аргумент функції повинен мати наступний вигляд **a=[3 2 -2].**

Функція повертає вектор всіх коренів алгебраїчного рівняння.

Розв'язання рівняння (3.10) функцією **roots** може мати наступний вигляд:

```
>> a=[2 20 -2 100];
>> roots(a)
ans =
 -10.5445 + 0.0000i
   0.2723 + 2.1605i
   0.2723 - 2.1605i
```

Результати повністю співпадають з результатами застосування функції **polyroots** СКМ MathCAD

Контрольні запитання

В яких виглядах записують рівняння?

Що називають коренями рівняння, що таке парні корені рівняння?

В чому відмінність прямих та ітераційних методів розв’язання рівнянь?

Назвіть типи ітераційних методів розв’язання рівнянь.

З яких етапів складається ітераційне розв’язання рівнянь?

Що є критерієм знаходження кореня?

Що таке ізоляція коренів?

До яких методів відноситься метод хорд?

В чому полягає особливість методу простої ітерації?

Які методи називають лінеаризованими?

Для розв'язання яких рівнянь призначено функцію **polyroots**?

Чим обумовлюється точність розв'язання рівнянь функцією **root**?

Як обрати метод "ділення навпіл" в функції **root**?

Назвіть способи описання рівняння для функції **fzero**?

Скільки вихідних параметрів має функція **fzero**?

Для чого використовується структура **Optimset** в функції **fzero**?

4. РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

Сукупність M функцій від N аргументів x, яка записана у вигляді

$$\begin{cases} f_1(x_1, x_2, \dots, x_N) = 0 \\ \vdots \\ f_M(x_1, x_2, \dots, x_N) = 0 \end{cases} \qquad \overrightarrow{f(\vec{x})} = 0, \tag{4.1}$$

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_N) \\ \vdots \\ x_M = \varphi_1(x_1, x_2, \dots, x_N) \end{cases} \qquad \vec{x} = \overrightarrow{\varphi(\vec{x})}. \tag{4.1'}$$

називається **системою рівнянь**.

Упорядкована сукупність N чисел, **x*1, x*2, …, x*N,** яка, будучи підставленою в систему замість **x1, x2, …, xN,** перетворює всі рівняння в правильні числові рівності, називається **розв'язком** системи.

В разі, коли кількість рівнянь та кількість невідомих співпадають (**M=N**), система називається **визначеною**. Коли кількість рівнянь більша за кількість невідомих (**M>N**), система називається **недовизначеною**, коли кількість рівнянь менша за кількість невідомих (**M<N**), система називається **перевизначеною**.

Якщо функції **fi** є алгебраїчними, систему називають **системою лінійних алгебраїчних рівнянь** (**СЛАР**):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2N}x_N = b_2 \\ \vdots \\ a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \cdots + a_{NN}x_N = b_N \end{cases},$$

або в скороченому вигляді:

$$\sum_{j=1}^{N} a_{ij}x_j = b_i \ (i = 1,2 \dots N). \tag{4.1''}$$

Індекс «i» вказує рівняння, індекс «j» – змінну.

Наприклад,

$$\begin{cases} 4x_1 + 2x_2 + x_3 = 1 \\ 3x_1 - 1.5x_2 + 6x_3 = 2 \\ x_1 + 3x_2 - 4x_3 = 3 \end{cases}.$$

Для розв’язування систем рівнянь, як і просто рівнянь, застосовують **точні** та **ітераційні** методи.

Точні методи розроблені тільки для розв'язування СЛАР. До точних належать метод Гауса, схема Холецького, метод квадратних коренів, правило Крамера тощо.

До **ітераційних** належать методи Ньютона, метод Зейделя тощо. **Ітераційні** методи можуть застосовуватися до будь-яких типів систем рівнянь.

В якості критерію досягнення точності **ітераційних** методів розв'язування систем рівнянь для абсолютної погрішності використовується «**жорстка»** норми вектора різниці між суміжними значеннями отриманих на певному кроці наближень

$$|\overrightarrow{x_{k+1}} - \overrightarrow{x_k}| < \varepsilon_x,$$

або «**м'яка»** поелементна перевірка абсолютного значення різниці між суміжними значеннями отриманих на певному кроці наближень

$$\max\left(\left|x_i^{(k+1)} - x_i^{(k)}\right|\right) < \varepsilon_x.$$

Систему (4.1") можна записати у вигляді одного матричного рівняння

$$\mathrm{A}\vec{x} = \overrightarrow{b}, \qquad (4.2)$$

де $\mathrm{A} = (a_{ij}) = \begin{pmatrix} a_{11}\, a_{12}\, ..a_{1n} \\ a_{21}\, a_{22}\, ..a_{2n} \\ ..... \\ a_{n1}\, a_{n2}\, ..a_{nn} \end{pmatrix}$ – матриця коефіцієнтів;

$\vec{b}=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$, $\vec{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ – відповідно стовпець вільних членів і стовпець змінних.

Якщо СЛАР (4.2) є визначеною, тобто має квадратну матрицю А коефіцієнтів, система має розв'язок. Якщо визначник системи (4.2) не дорівнює нулю

$$\Delta = \det(A) = \begin{vmatrix} a_{11}\, a_{12} \ldots a_{1n} \\ a_{21}\, a_{22} \ldots a_{2n} \\ \ldots\ldots\ldots \\ a_{n1}\, a_{n2} \ldots a_{nn} \end{vmatrix} \neq 0,$$

то вона має єдиний розв’язок.

Найбільш уживаний запис для розв'язання СЛАР записується як

$$\vec{x} = A^{-1}\overrightarrow{b},$$

де $\vec{x}$ – шуканий вектор коренів; A^{-1} – обернена матриця; $\vec{b}$ – вектор вільних членів.

Недоліком **прямого** розв'язання СЛАР є необхідність виконання великої кількості обчислювальних операцій для розрахунку оберненої матриці.

$$A^{-1} = \frac{Adj(A)^{\mathrm{T}}}{\det(A)}\ i,k = 1 \ldots N,$$

де **Adj (A)** – приєднана матриця, елементами якої є алгебраїчні доповнення матриці **A**; $\mathbf{M_{ik}}$ – мінор елемента **ik**.

Визначник матриці А та приєднана матриця розраховуються як

$$\det(A) = \sum_{k=1}^{N} a_{ik}M_{ik}\ \ (i = const)\ \ = \sum_{i=1}^{N} a_{ik}M_{ik}\ \ (k = const),$$

$$Adj(A) = \begin{bmatrix} M_{11} & \ldots & (-1)^{1+N}M_{1N} \\ . & (-1)^{i+k}M_{ik} & . \\ (-1)^{N+1}M_{N1} & \ldots & (-1)^{N+N}M_{NN} \end{bmatrix}.$$

Для усунення необхідності розрахунку оберненої матриці при розв'язання СЛАР може використовуватися метод Крамера

$$x_k = \frac{\det(A_k)}{\det(A)}, \qquad k = 1,2,\dots,N,$$

де матриця $A_к$ є матрицею А, в якій к-й стовпець замінений стовпцем вільних членів.

Метод Гауса (послідовного виключення змінних, подвійного ходу)

Найпростішим методом розв'язання СЛАР без обрахунку оберненої матриці та визначника матриці коефіцієнтів є метод послідовного виключення змінних, або метод Гауса.

Систему розв'язують в два етапи. На першому етапі «прямого ходу» вихідну матрицю **A** зводять до рівносильної їй верхньої трикутної матриці.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & a_{23} & a_{2N-1} & a_{2N} \\ a_{31} & a_{32} & a_{33} & a_{3N-1} & a_{3N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N-11} & a_{N-12} & a_{N-13} & a_{N-1N-1} & a_{N-1N} \\ a_{N1} & a_{N2} & a_{N3} & a_{NN-1} & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{N-1} \\ b_N \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 1 & a'_{12} & a'_{13} & a'_{1N-1} & a'_{1N} \\ 0 & 1 & a'_{23} & a'_{2N-1} & a'_{2N} \\ 0 & 0 & 1 & a'_{3N-1} & a'_{3N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & a'_{N-1N} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_{N-1} \\ b'_N \end{bmatrix},$$

$$\begin{cases} 1x_1 + a'_{12}x_2 + \cdots + a'_{1N}x_N = b'_1 \\ 0 + 1x_2 + a'_{23}x_3 + \cdots + a'_{2N}x_N = b'_2 \\ \vdots \\ 1x_{N-2} + a'_{N-2N-1}x_{N-1} + a'_{N-2N}x_N = b'_{N-2}. \\ 1x_{N-1} + a'_{N-1N}x_N = b'_{N-1} \\ 1x_N = b'_N \end{cases} \qquad (4.3)$$

На другому етапі «зворотного ходу» знаходять розв'язок СЛАР.

«Зворотний» хід

З останнього рівняння (4.3) можливо визначити x_N

$$x_N=b'_N.$$

З урахуванням відомого x_N з передостаннього рівняння можна визначити x_{N-1}

$$x_{N-1}=b'_{N-1}-a'_N-N\cdot x_N.$$

Аналогічно за результатами двох останніх виразів можна визначити корінь x_{N-2}

$$x_{N-2}=b'_{N-2}-(a'_N-2N\cdot x_N+ a'_N-2N-x_{N-1}).$$

Далі, піднімаючись вгору, кожний і-й корінь і-го рівняння може бути розрахований через відомі корені рівнянь з і-го по останнє.

$$x_i = b'_i - \sum_{k=i+1}^{N} a_{ik}x_k. \qquad (4.4)$$

Вираз (4.4) є правилом «**зворотного**» ходу.

«Прямий» хід

Дії **прямого** ходу виконуються за два кроки послідовно для всіх рівнянь системи, тобто рядків матриці коефіцієнтів та вектора вільних членів. На першому кроці робочий рядок приводиться до вигляду, коли на елемент головної діагоналі дорівнює 1, шляхом ділення всіх елементів рядка на елемент головної діагоналі. На другому кроці робляться перетворення, які забезпечують, щоб всі елементи стовпця з номером робочого рядка для всіх рядків, які розташовуються нижче робочого, дорівнювали 0. Тобто з усіх рівнянь, нижче робочого виключається змінна з номером робочого рядка шляхом віднімання від відповідного рівняння робочого рівняння, яке помножене на значення елемента з рядка, який обнуляється.

Перший цикл. Робочим рядком є перший рядок

$$\begin{bmatrix} 1 & a'_{12} & a'_{13} & a'_{1N} \\ 0 & a'_{22} & a'_{23} & a'_{2N} \\ 0 & a'_{32} & a'_{33} & a'_{3N} \\ 0 & a'_{N2} & a'_{N3} & a'_{NN} \end{bmatrix} \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_N \end{bmatrix}.$$

Перший крок.

$$a'_{11}=a_{11}/a_{11}=1 \quad a'_{12}=a_{12}/a_{11} \quad \ldots \quad a'_{1N}=a_{1N}/a_{11} \quad b'_1=b_1/a_{11}.$$

Другий крок.

$a'_{21}=a_{21}-a_{21}a'_{11}$ $a'_{22}=a_{22}-a_{21}a'_{12}$ $a'_{23}=a_{23}-a_{21}a'_{13}$ ... $a'_{2N}=a_{2N}-a_{21}a'_{1N}$ $b'_2=b_2-a_{21}b'_1$

$a'_{31}=a_{31}-a_{31}a'_{11}$ $a'_{32}=a_{32}-a_{31}a'_{12}$ $a'_{23}=a_{33}-a_{31}a'_{13}$ ... $a'_{3N}=a_{3N}-a_{31}a'_{1N}$ $b'_3=b_3-a_{31}b'_1$

$a'_{N1}=a_{N1}-a_{N1}a'_{11}$ $a'_{N2}=a_{N2}-a_{N1}a'_{12}$ $a'_{N3}=a_{N3}-a_{N1}a'_{13}$ ... $a'_{NN}=a_{NN}-a_{N1}a'_{1N}$ $b'_N=b_N-a_{N1}b'_1$.

Другий цикл.

Робочим рядком є другий рядок

$$\begin{bmatrix} 1 & a'_{12} & a'_{13} & a'_{1N} \\ 0 & 1 & a'_{23} & a'_{2N} \\ 0 & 0 & a'_{33} & a'_{3N} \\ 0 & 0 & a'_{N3} & a'_{NN} \end{bmatrix} \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_N \end{bmatrix}.$$

Перший крок.

$a'_{21}=a'_{21}/a_{22}$ $a'_{22}=a'_{22}/a'_{22}$ ... $a'_{2N}=a'_{2N}/a'_{22}$ $b'_2=b'_2/a'_{22}$.

Другий крок.

$a'_{31}=a'_{31}-a'_{32}a'_{21}$ $a'_{32}=a'_{32}-a'_{32}a'_{22}$ $a'_{33}=a'_{33}-a'_{32}a'_{23}$ ... $a'_{3N}=a'_{3N}-a'_{32}a'_{2N}$ $b'_3=b'_3-a'_{32}b'_2$

$a'_{41}=a'_{41}-a'_{42}a'_{21}$ $a'_{42}=a'_{42}-a'_{42}a'_{22}$ $a'_{43}=a'_{43}-a'_{42}a'_{13}$ ... $a'_{4N}=a'_{4N}-a'_{42}a'_{2N}$ $b'_4=b_4-a_{42}b'_2$

$a'_{N1}=a'_{N1}-a'_{N2}a'_{21}$ $a'_{N2}=a'_{N2}-a'_{N2}a'_{22}$ $a'_{N3}=a'_{N3}-a'_{N2}a'_{23}$. $a'_{NN}=a'_{NN}-a'_{N2}a'_{1N}$ $b'_N=b'_N-a'_{N2}b'_2$.

Третій цикл. Робочим рядком є третій рядок

$$\begin{bmatrix} 1 & a'_{12} & a'_{13} & a'_{1N} \\ 0 & 1 & a'_{23} & a'_{2N} \\ 0 & 0 & 1 & a'_{3N} \\ 0 & 0 & 0 & a'_{NN} \end{bmatrix} \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_N \end{bmatrix}.$$

Перший крок.

$a'_{31}=a'_{31}/a'_{33}$ $a'_{32}=a'_{32}/a'_{32}$ ... $a'_{3N}=a'_{3N}/a'_{33}$ $b'_3=b'_3/a'_{33}$.

Другий крок.

$a'_{41}=a'_{41}-a'_{43}a'_{31}$ $a'_{42}=a'_{42}-a'_{43}a'_{32}$ $a'_{43}=a'_{43}-a'_{43}a'_{33}$ ... $a'_{4N}=a'_{4N}-a'_{43}a'_{3N}$ $b'_4=b'_4-a'_{43}b'_3$

$a'_{N1}=a'_{N1}-a'_{N3}a'31$ $a'_{N2}=a'_{N2}-a'_{N3}a'_{32}$ $a'_{N3}=a'_{N3}-a_{N3}a'_{23}$ ... $a'_{NN}=a'_{NN}-a'_{N3}a'_{1N}$ $b'_N=b_N-a'_{N3}b'_3$.

N – й цикл завершить прямий хід з результуючою верхньою трикутною матрицею коефіцієнтів.

4.1. Засоби розв'язання СЛАР СКМ MathCAD, Matlab

Засоби MathCAD

Для визначених систем, в яких кількість рівнянь співпадає з кількістю невідомих може застосовуватися «прямий», матричний метод розв’язання системи лінійних рівнянь (СЛАР)

$$\vec{x} = A^{-1}\vec{b}.$$

Розв'язання всіх СЛАУ, в тому числі перевизначених та невизначених систем, в MathCAD забезпечує функція **lsolve**. В функції реалізовано різновид **метода Гауса** – метод **трикутних LU матриць**.

Похибка розв’язання СЛАР «прямим» методом та функцією **lsolve** не залежить від значення змінної **TOL** та збільшується зі збільшенням розмірності системи.

Точність розв’язання СЛАР в значній мірі залежить від **обумовленості** системи. **Мірою обумовленості** СЛАР слугує **число обумовленості** (**condition number**). Системи, в яких коефіцієнти кількох рівнянь близькі називаються системами з **поганою обумовленістю**. Погана обумовленість призводить до нестійкості методів розв’язання та збільшенню похибки. Оцінити числа обумовленості в MathCAD можна функціями **cond1, cond2, conde**. В разі отримання невірної відповіді слід перевірити коректність СЛАР.

В якості аргументів функції **lsolve(A,B)** визначається матриця коефіцієнтів СЛАР A та вектор-стовпець правих частин СЛАР B.

В наведеному нижче прикладі функція **lsolve** застосована для перевизначеної (3x2) та невизначеної (3x4) СЛАР.

З результатів прикладу видно, що для визначених систем матричний метод та функція **lsolve** забезпечують вірне розв’язання та надають однаковий результат. Для систем з неквадратною матрицею матричний метод не дозволяє знайти розв’язання, а функція **lsolve** знаходить розв'язок, який є невірним для недовизначної системи та вірним – для перевизначеної.

$$A := \begin{pmatrix} 2.3 & 6.7 & 9.8 \\ 24 & 42 & 1 \\ 6 & 0 & 12 \end{pmatrix} \qquad B := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$x1 := A^{-1} \cdot B = \begin{pmatrix} 0.272 \\ -0.111 \\ 0.114 \end{pmatrix} \qquad x01 := \text{lsolve}(A, B) = \begin{pmatrix} 0.272 \\ -0.111 \\ 0.114 \end{pmatrix} \qquad A \cdot x01 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$A := \begin{pmatrix} 2.3 & 9.8 \\ 24 & 1 \\ 6 & 12 \end{pmatrix} \quad x2 := A^{-1} \cdot B = \quad x02 := \text{lsolve}(A, B) = \begin{pmatrix} 0.08 \\ 0.159 \end{pmatrix} \quad A \cdot x02 = \begin{pmatrix} 1.742 \\ 2.082 \\ 2.388 \end{pmatrix}$$

$$A := \begin{pmatrix} 2.3 & 6.7 & 2.8 & 9.8 \\ 24 & 42 & 7 & 1 \\ 6 & 0 & 8 & 12 \end{pmatrix} \quad x3 := A^{-1} \cdot B = \quad x03 := \text{lsolve}(A, B) = \begin{pmatrix} 0.156 \\ -0.069 \\ 0.155 \\ 0.068 \end{pmatrix}$$

$$A \cdot x03 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad +$$

Отримані вбудованими засобами розв'язку СЛАР ОБОВ'ЯЗКОВО потребують перевірки.

Навіть для визначених СЛАР MathCAD без попередження можуть видати невірну відповідь. Наприклад, для наведеної нижче визначеної системи 4x4 обидва методи не дали вірної відповіді. Причина полягає в тому, що наведена СЛАР не може розв'язуватися стандартними методами, так як визначник матриці дорівнює нулю.

$$A := \begin{pmatrix} 1 & 2 & -3.2 & 3.4 \\ 2 & 1 & -0.2 & -0.4 \\ 3 & 1.4 & -0.3 & -0.6 \\ -5 & -10 & 16 & -17 \end{pmatrix} \qquad B := \begin{pmatrix} 1.2 \\ 1.8 \\ 2.7 \\ 6 \end{pmatrix}$$

$$|A| = 0 \qquad \text{lsolve}(A,B) = \begin{pmatrix} -1.708 \times 10^{15} \\ 26.457 \\ -6.276 \times 10^{15} \\ -5.404 \times 10^{15} \end{pmatrix} \qquad A^{-1} \cdot B = \begin{pmatrix} -1.708 \times 10^{15} \\ 24 \\ -6.276 \times 10^{15} \\ -5.404 \times 10^{15} \end{pmatrix}$$

Засоби Matlab

СКМ Matlab має набагато багатшу номенклатуру функцій, які можуть бути застосовні для розв'язання СЛАР. Аргументами цих функцій слугують матриця коефіцієнтів СЛАР A та вектор вільних членів СЛАР B.

Розв'язок обумовлених СЛАР може бути знайдений функціями правого «/» та лівого «\» ділення.

Праве ділення **`X=A/B`** призначено для розв'язання СЛАР **`XA=B`**, де B та X – вектори-рядки.

Ліве ділення **`X=A\B`** призначено для розв'язання СЛАР **`AX=B`**, де B та X – вектори-стовпці. Обумовлені СЛАР розв'язується **методом Гауса**. В інших випадках записом **`X=A\B`** система розв'язується **методом мінімізації нев'язок другої норми** матриці.

Matlab має спеціалізовані засоби для розв'язання особливих СЛАР. Засоби базуються на **методі найменших квадратів**.

Для розв'язання СЛАР з особливою правою частиною **`A*X=B+e`** призначено функцію **`[x<,stdx,mse,S>]=lscov(A,B,V,<alg>).`** Аргументами функції слугують матриця коефіцієнтів СЛАР **`A`** розміром **`mxXxn`**, де **`m>n`**. вектор вільних членів СЛАР **`B`** та коваріаційна матриця **`V`** вектора **`e`**. Опційно може бути визначено алгоритм **`alg`**: **`'chol'`** – схема Холецького, або **`'orth'`** – ортогональне перетворення В якості критерію мінімізації використовується умова **`(AX-b)'*inv(V)*(AX-b).`**

Розв’язок СЛАР шукається за наступним правилом **X=inv(A'*inv(V)*A)*A'*inv(V)*B**. Інвертування коваріаційної матриці не проводиться.

Для додаткового аналізу результатів розв'язання в векторі результату крім вектора коренів **X** можна отримати значення середньоквадратичного відхилення **stdx**, середньоквадратичної похибки **mse** та коваріаційної матриці вектора коренів **S:**

```
mse=B'*(inv(V)-inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))...
*B./(m-n)

S = inv(A'*inv(V)*A)*mse

stdx = sqrt(diag(S))
```

Для розв’язання СЛАР з розрідженою матрицею **A** мінімізацією норми ||**b-A*x**|| з наперед визначеною точністю **tol** призначена функція **lsqr(A,B<,tol>).**

Для додаткового аналізу результатів розв'язання в векторі результату крім вектора коренів **X** можна отримати ознаку закінчення обчислень **flag**, відносне значення норми **relres=||(b-A*x)||/||norm(b)||**, значення кількості ітерацій **iter** тощо.

Функція **pcg(A,B)** розв’язує СЛАР з квадратною додатно визначеною матрицею A **методом спряжених градієнтів**.

Функція **cgs(A,B)** реалізує **квадратичний метод спряжених градієнтів**.

Двонаправлений метод спряжених градієнтів та стійкий двонаправлений метод спряжених градієнтів розв'язання СЛАР реалізовані функцією **bicg(A,B)** та **bicg(A,B)** відповідно.

Критерієм для функцій **lsqr, bicg, pcg, cgs** слугує значення відносної норми СЛАР **norm(B-Ax)/norm(B) < tol** (за замовчанням 10^{-6}).

Для випадку, коли необхідно провести розв’язання СЛАР для однієї **nxXxn** матриці A з декількома варіантами вектора B можна скористатися **розкладанням Холецького** функцією **[R/L<,p,S>]=chol(A<,mode>).** Функція повертає праву/ліву **R/L** трикутні матриці розкладання в залежності

від значення режиму **mode** – **'upper'/'lower'** відповідно, матрицю перестановок **S**.

R – верхня трикутна матриці така, що **R'*R=S'*A*S**, для p=0. Для інших значень **p, R** є верхньою трикутною матрицею розміру **qxXxn**. Застосування схеми дає виграш приблизно в n разів. Система **A*x=B** переводиться в систему **R'*R*x=B** з розв'язком **x=R\(R'\B).**

Контрольні запитання

В яких формах записують системи рівнянь?

Що називають коренем системи рівнянь?

Що таке визначена, недовизначена, перевизначена система рівнянь?

Що є умовою можливості розв'язання СЛАР?

В чосу полягає недолік матричного розв'язання СЛАР?

З яких етапів складається метод Гауса?

Для чого призначена функція **fsolve**?

Як оцінити обумовленість матриці?

В чому полягає різниця між діями "\" та "/" в СКМ Matlab?

На якому методі базуються спеціальні засоби розв'язання СЛАР в СКМ Matlab?

Які методи реалізовані для розв'язання СЛАР в СКМ Matlab?

5. ЗАСОБИ РОЗВ'ЯЗАННЯ СИСТЕМ НЕЛІНІЙНИХ РІВНЯНЬ

Для розв'язання систем нелінійних рівнянь використовуються **ітераційні** методи.

Метод простої ітерації

Метод є розширенням методу простої ітерації для розв'язання рівнянь на систему рівнянь, яка записана в явному вигляді

$$\vec{x} = \overrightarrow{\varphi(\vec{x})} \quad \begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_N) \\ \cdots \\ x_N = \varphi_N(x_1, x_2, \dots, x_N) \end{cases}. \tag{5.1}$$

Процес простої ітерації для системи нелінійних рівнянь збігається до її розв'язку, якщо норма **φ'(x)** в заданому околі є менша за одиницю, тобто для збіжності розв'язків системи достатньою є умова

$$\left\| \overrightarrow{\varphi'(\vec{x})} \right\| < 1, \tag{5.2}$$

де φ' матриця частинних похідних, яку називають матрицею Якобі:

$$\overrightarrow{\varphi(\vec{x})'} = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_N}{\partial x_1} & \cdots & \frac{\partial \varphi_N}{\partial x_1} \end{pmatrix}.$$

Схема методу

1. Вихідним є запис системи в явному вигляді (5.1), визначення точності **εx**, вектора початкових значень коренів

$$\overrightarrow{x^{(0)}} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_N^{(0)} \end{bmatrix}. \tag{5.3}$$

2. Підставляння вихідного вектора (5.3) в систему (5.1). Послідовне обчислення наближення першої змінної з першого рівняння

системи, другої змінної – з другого рівняння системи тощо. Ітераційне правило має наступний вигляд

$$\begin{cases} x_1^{(1)} = \varphi_1(x_1^{(0)}, x_2^{(0)}, \dots, x_N^{(0)}) \\ \vdots \\ x_N^{(1)} = \varphi_N(x_1^{(0)}, x_2^{(0)}, \dots, x_N^{(0)}) \end{cases}.$$

3. Перевірка точності. Якщо точність досягнута, то значення вектора наближення є розв'язанням системи, якщо точність не досягнута, то початкове значення замінюється на значення вектора наближення $\overrightarrow{x^{(0)}} = \overrightarrow{x^{(1)}}$ та повторюється п.2.

Найчастіше метод простої ітерації використовується у вигляді **модифікації Зейделя**. Сутність модифікації полягає у використовуванні уточнених значень змінних уже на поточному ітераційному кроці. Так, для уточнення на (к+1)-му кроці значення першої змінної x_1 використовуються значення попереднього (к)-го кроку, для другої змінної – значення **x1** (к+1)-го кроку, а для решти змінних - значення попереднього (к)-го кроку. Ітераційний алгоритм має наступний вигляд:

$$\begin{gathered} x_1^1 = \varphi_1(x_1^0, x_2^0, \dots, x_N^0) \\ x_2^1 = \varphi_2(x_1^1, x_2^0, \dots, x_N^0) \\ x_3^1 = \varphi_2(x_1^1, x_2^1, x_3^0, \dots, x_N^0) \\ \vdots \\ x_N^1 = \varphi_N(x_1^1, x_2^1, \dots, x_{N-1}^1, x_N^0) \end{gathered} .$$

Умова завершення процесу розв'язання системи нелінійних рівнянь за **модифікацією Зейделя** збігається з умовою для методу простих ітерацій.

Метод Ньютона

Метод є розширенням методу дотичних для розв'язання рівнянь на системи рівнянь, яка записана в нормальному вигляді

$$\overrightarrow{f(\vec{x}) = 0} \quad \begin{cases} f_1(x_1, x_2, \dots, x_N) = 0 \\ \cdots \\ f_N(x_1, x_2, \dots, x_N) - 0 \end{cases}. \qquad (5.4)$$

Процес ітерації для методу дотичних системи нелінійних рівнянь збігається до її розв'язку, якщо визначник матриці Якобі системи не дорівнює нулю.

Математичним підґрунтям методу є лінеаризація функцій $\overrightarrow{f(\vec{x})}$ шляхом розкладання в ряд Тейлора в околі точки початкового значення $\overrightarrow{x^{(0)}}$ й відкидання всіх членів ряду, окрім лінійних щодо приростів змінних.

Для функцій N змінних лінійна частка розкладання в ряд Тейлора в околі точки $x^{(0)}$:

$$f_i(\vec{x}) = f_i\left(\overrightarrow{x^{(0)}}\right) + \left(x_1 - x_1^{(0)}\right)\frac{\partial f_i\left(\overrightarrow{x^{(0)}}\right)}{\partial x_1} + \left(x_2 - x_2^{(0)}\right)\frac{\partial f_i\left(\overrightarrow{x^{(0)}}\right)}{\partial x_2} +$$

$$\ldots \qquad + \left(x_N - x_N^{(0)}\right)\frac{\partial f_i\left(\overrightarrow{x^{(0)}}\right)}{\partial x_N} = f_i\left(\overrightarrow{x^{(0)}}\right) + \sum_{j=1}^{N}\left(x_j - x_j^{(0)}\right)\frac{\partial f_i\left(\overrightarrow{x^{(0)}}\right)}{\partial x_j}. \quad (5.5)$$

З урахуванням (5.5) система (5.4) може бути записана в наступному вигляді

$$\begin{cases} f_1(\vec{x}) - f_1\left(\overrightarrow{x^{(0)}}\right) = \dfrac{\partial f_1\left(\overrightarrow{x^{(0)}}\right)}{\partial x_1}(x_1 - x_1^{(0)}) + \dfrac{\partial f_1\left(\overrightarrow{x^{(0)}}\right)}{\partial x_2}(x_2 - x_2^{(0)}) + \cdots + \dfrac{\partial f_1\left(\overrightarrow{x^{(0)}}\right)}{\partial x_N}(x_N - x_N^{(0)}) \\ \cdots \\ f_N(\vec{x}) - f_N\left(\overrightarrow{x^{(0)}}\right) = \dfrac{\partial f_N\left(\overrightarrow{x^{(0)}}\right)}{\partial x_1}(x_1 - x_1^{(0)}) + \dfrac{\partial f_N\left(\overrightarrow{x^{(0)}}\right)}{\partial x_2}(x_2 - x_2^{(0)}) + \cdots + \dfrac{\partial f_N\left(\overrightarrow{x^{(0)}}\right)}{\partial x_N}(x_N - x_N^{(0)}) \end{cases},$$

В точці кореня значення функції рівняння дорівнює нулю. Відповідно, систему (5.4) можна записати в наступному вигляді:

$$\begin{cases} \Delta F_{11}\Delta x_1 + \Delta F_{12}\Delta x_2 + \cdots + \Delta F_{1N}\Delta x_N = -f_1\left(\overrightarrow{x^{(0)}}\right) \\ \cdots \\ \Delta F_N\Delta x_1 + \Delta F_{N2}\Delta x_2 + \cdots + \Delta F_{NN}\Delta x_N = -f_N\left(\overrightarrow{x^{(0)}}\right) \end{cases}, \quad (5.6)$$

де $\Delta x_i = x_i - x_i^{(0)}$ – приріст і-ї змінної; $\Delta F_{ij} = \frac{\partial f_i(\overrightarrow{x^{(0)}})}{\partial x_j}$ – частинна похідна і-ї функції системи по j-й змінній, яка обчислена в точці початкового наближення.

В матричному вигляді запис виглядатиме як

$$\begin{bmatrix} \frac{\partial f_1(\overrightarrow{x^{(0)}})}{\partial x_1} & \cdots & \frac{\partial f_1(\overrightarrow{x^{(0)}})}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(\overrightarrow{x^{(0)}})}{\partial x_1} & \cdots & \frac{\partial f_N(\overrightarrow{x^{(0)}})}{\partial x_1} \end{bmatrix} \begin{bmatrix} x_1 - x_1^{(0)} \\ \vdots \\ x_N - x_N^{(0)} \end{bmatrix} = \begin{bmatrix} -f_1(\overrightarrow{x^{(0)}}) \\ \vdots \\ -f_N(\overrightarrow{x^{(0)}}) \end{bmatrix}. \quad (5.6')$$

Вираз (5.6') є СЛАР порядку **N** відносно приросту змінних

$$\overrightarrow{\Delta x} = -\overrightarrow{f\left(\overrightarrow{x^{(0)}}\right)}. \qquad (5.6'')$$

Розв'язок СЛАР за умови, що **det(ΔF)≠0** надає вектор відхилень

$$\overrightarrow{\Delta x} = -\Delta F^{-1}\overrightarrow{f(\overrightarrow{x^{(0)}})}.$$

Нове наближення знаходиться додаванням вектора відхилень до вектора початкових значень

$$\vec{x} = \overrightarrow{x^{(0)}} + \overrightarrow{\Delta x}. \qquad (5.7)$$

Схема методу

1. Визначення системи в нормальному вигляді (5.4), точності **εx**, вектора початкових значень коренів
$$\overrightarrow{x^{(0)}} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_N^{(0)} \end{bmatrix}.$$
2. Обчислення Якобіану ΔF при початкових значеннях. Розв'язання СЛАР (5.6'). Знаходження нового наближення (5.7).
3. Перевірка досягнення точності. В разі виконання умови точності знайдене наближення є розв'язком системи. В разі невиконання умови провести зміну початкового наближення знайденим $\overrightarrow{x^{(0)}} = \vec{x}$ та повторити процес з п.2.

Приклад. Розв'язати систему рівнянь

$$\begin{cases} 2x^2 - 5x - xy + 1 = 0 \\ x + 3log(x) - y^2 = 0 \end{cases}.$$

РОЗВ'ЯЗОК

Аналіз графіків (рис.5.1) показує наявність двох розв'язків: в околі точок (1.5 -2) та (3.5 2).

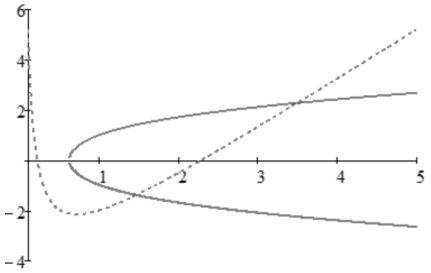


Рисунок 5.1 – Графік функцій системи

За початкове наближення обираємо точку (3 2).

Правило переведення рівняння з нормального виду до явного не призначено для системи рівнянь, тому переводимо систему в явний вигляд як

$$\begin{cases} y = \dfrac{2x^2 - 5x + 1}{x} \\ x = y^2 - 3\log(x) \end{cases}.$$

Якобіан системи для методу простих ітерацій становить

$$\begin{pmatrix} \dfrac{4x-5}{x} - \dfrac{2x^2-5x+1}{x^2} & 0 \\ -\dfrac{3}{xln(10)} & 2y \end{pmatrix}.$$

В початковій точці (3 2) Якобіан становить

$$\begin{pmatrix} 1.89 & 0 \\ -0.43 & 4 \end{pmatrix}, \qquad \|\varphi'(x0)\|=4.03.$$

умови застосування методу простих ітерацій не виконуються по обох рівняннях.

Змінимо вигляд системи

$$\begin{cases} x = \dfrac{2x^2 - 5x + 1}{y} \\ y = \sqrt{x + 3\log(x)} \end{cases}.$$

Якобіан системи становить

$$\begin{pmatrix} \dfrac{4x-5}{y} & \dfrac{2x^2-5x+1}{y^2} \\ \dfrac{\dfrac{3}{xln(10)} + 1}{2\sqrt{x + 3\log(x)}} & 0 \end{pmatrix}.$$

В початковій точці (3 2)

$$\begin{pmatrix} 3.51 & -1 \\ 0.34 & 0 \end{pmatrix}, \qquad \|\varphi'(x0)\|=3.66.$$

Результат покращився, але умови не виконуються по першому рівнянню.

Змінимо перше рівняння

$$\begin{cases} x = \sqrt{\dfrac{xy + 5x - 1}{2}} \\ y = \sqrt{x + 3\log(x)} \end{cases}.$$

Якобіан системи становить

$$\begin{pmatrix} \dfrac{y+5}{\sqrt{8(5x+xy-1)}} & \dfrac{x}{\sqrt{8(5x+xy-1)}} \\ \dfrac{\dfrac{3}{xln(10)}+1}{2\sqrt{x+3\log(x)}} & 0 \end{pmatrix}.$$

В початковій точці (3 2)

$$\begin{pmatrix} 0.55 & 0.24 \\ 0.34 & 0 \end{pmatrix}, \qquad \|\varphi'(x0)\|=0.68.$$

Умови виконуються.

Результати ітераційних циклів розв'язання системи методом простих ітерацій, Зейделя, Ньютона наведено в таблиці 5.1.

Таблиці 5.1 – Результати ітерацій розв'язання СНР

	Простих ітерацій				Зейделя				Дотичних			
	x	y	Δx	Δy	x	y	Δx	Δy	x	y	Δx	Δy
	3	2			3	2			3	2		
1	3.162	2.105	0.162	0.105	3.162	2.159	0.162	0.159	3.592	2.32	0.592	0.32
2	3.276	2.159	0.114	0.054	3.289	2.2	0.127	0.041	3.491	2.263	0.101	0.057
3	3.351	2.196	0.075	0.037	3.368	2.225	0.079	0.025	3.487	2.262	0.004	0.001
4	3.4	2.22	0.049	0.024	3.422	2.242	0.054	0.017	3.487	2.262		
5	3.431	2.235	0.031	0.015	3.448	2.25	0.026	0.008				

З наведених результатів очевидною є перевага збіжності методу дотичних.

5.1. Засоби розв’язання СНР СКМ MathCAD, Matlab

Засоби MathCAD

Чисельне розв'язання систем рівнянь у пакеті проводиться блоковим оператором **given-find (given-minerr**).

Для застосування блока **given-find** обов'язковим є попереднє визначення первинних значень змінних, для яких проводиться обрахунок системи. Обчислювальний блок виділяється згори службовим словом **given,** знизу – службовим словом **find(x1 <,x2,...>)/minerr(x1 <,x2,...>),** між якими записується необхідна система. В якості аргументів службового слова **find/minerr** зазначаються імена змінних, відносно яких потребується провести розв’язання. Обчислювальний блок знаходить вектор-рядок ДІЙСНОГО розв'язання системи.

Обчислювальні блоки не можуть бути розташовані один всередині другого.

Для запису рівнянь системи не може застосовуватися оператор прирівнювання «**:=**» та ранжовані змінні. Рівняння системи повинні записуватися через знак логічного «жирного» рівняння (Ctrl +=) «**=**».

Для розв'язання системи нерівностей замість знаків логічного рівняння слід поставити знаки нерівності (<, >). Відповіддю буде значення координат однієї точки, яка задовольняє умовам системи.

Для отримання координат іншої точки треба змінити початкові умови.

В якості чисельного методу при розв'язанні СНР обчислювальним блоком можуть призначатися лінійні або квадратичні методи. За замовчанням використовується **модифікований метод Ньютона**: **Левенберга-Маркардта**.

Контекстне меню функцій **Find, Minerr** дозволяє налаштувати режим розв’язання СНР (рис. 5.2).

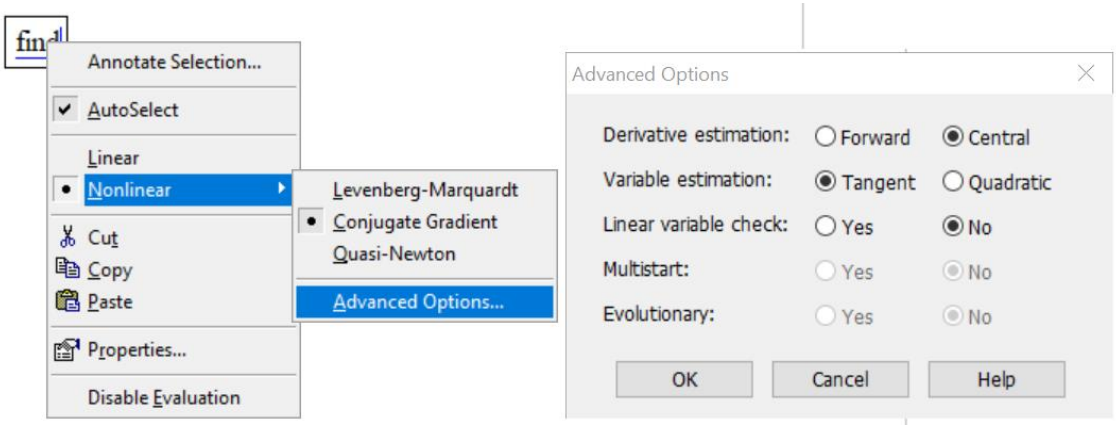


Рисунок 5.2 – Контекстне меню

Користувач може обрати **Linear** – симплекс метод першого порядку або, **Nonlinear** – квадратичні градієнтні методи: **Nonlinear Conjuctive Gradient** – метод спряжених градієнтів, **Quasi-Newton** – квазіньютонівський метод, **Levenberg-Marquardt** – метод Левенберга. Всі квадратичні методи є модифікованими методами Ньютона.

Пунктом **Advanced Options** відкриває відповідне вікно, в якому за потреби можна визначити способи обчислення похідних та наближення точок.

У вікні **Advanced Options** знаходяться поля **Derivative Estimation, Variable Estimation, Linear variable check, Multistart**, **Evolutionary.**

Поле **Derivative Estimation** визначає спосіб чисельного диференціювання: **Forward** – однобічне двоточкове, **Central** – центральне трьохточкове.

Поле **Variable Estimation** визначає спосіб оцінювання змінної: **Tangent** – дотичною прямою, **Quadratic** – параболою.

Поле **Linear variable check (Перевірка лінійності)** Вказує, чи треба на кожному кроці перевіряти часткових похідних функції (**Yes**), чи за умови невеликого приросту часткових похідних функції можна надати їм сталого значення та не перераховувати на кожній ітерації для скорочення часу (**No**)

Поля **Multistart**, **Evolutionary** призначені для застосування з функціями **Maximize**, **Minimize**. Вони дозволяють розширити область

пошуку точки оптимуму та знизити чутливість градієнтних методів від положення точки початку пошуку.

Розробник не рекомендує змінювати налаштування вікна **Advanced Options** без нагальної потреби.

Наприклад, розглянемо застосування обчислювального блока для аналізу взаємного розташування еліпса

$$\frac{x^2}{4}+\frac{y^2}{9}=25$$

та двох прямих

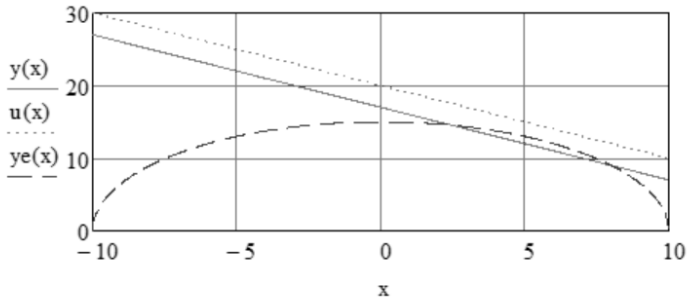
$$y(x)=17-x$$

$$u(x)=20-x.$$

Визначимо функції користувача та побудуємо графік еліпса та прямих:

$$f(x,y):=\frac{x^2}{4}+\frac{y^2}{9}-25 \qquad y(x):=17-x \qquad u(x):=20-x$$

$$ye(x):=3\cdot\sqrt{25-\frac{x^2}{4}}$$



З графіка видно, що пряма **y(x)=17-x** має дві точки перетину, в околі **x=2.5** та **x=8**. Друга пряма **u(x)=20-x** немає точок перетину з еліпсом.

Визначимо початкові значення та використаємо обчислювальний блок:

$$x:=4 \qquad y:=10$$

$$\text{Given} \qquad f(x,y)=0 \quad y+x=17 \qquad z:=\text{Find}(x,y)=\begin{pmatrix}8\\9\end{pmatrix}$$

$$\text{Given} \qquad f(x,y)=0 \quad y+x=20 \qquad z:=\text{Find}(x,y)=$$

Для початкових значень **x=4, y=10** блок **given-find** визначив точку перетину еліпса з прямою **y(x)** як корінь для першої системи та не

виявив точку перетину еліпса з прямою **u(x)**, бо не знайшов корінь для другої.

Для знаходження другої точки перетину еліпса з прямою **y(x)** слід або змінити початкові значення, або додати умову знаходження кореня в визначеному діапазоні. Діапазон пошуку можна визначити логічною умовою всередині обчислювального блоку:

$x := 4 \quad y := 10$

Given $\quad f(x,y) = 0 \quad y + x = 17 \quad x < 5 \quad z := \text{Find}(x,y) = \begin{pmatrix} 2.462 \\ 14.538 \end{pmatrix}$

$x := 2$

Given $\quad f(x,y) = 0 \quad y + x = 17 \quad z := \text{Find}(x,y) = \begin{pmatrix} 2.462 \\ 14.538 \end{pmatrix}$

Як видно з наведеного прикладу обидва способи дозволили отримати координати другої точки перетину еліпса та прямої **y(x)**.

Обчислювальний блок **given-find** не дозволяє знаходити корені несумісних систем функцію.

Для отримання не точних, а наближених значень коренів, можна застосувати в обчислювальному блоці замість функції **find** функцію **minerr**, яка використовує для розв'язання СНР метод **мінімізіції середньоквадратичних відхилень** та знаходить найбільш наближене розв’язання.

Для оцінки похибки розв'язання СНР функцією **minerr** можна використовувати вбудовану змінну **ERR**, яка містить середньоквадратичне значення (RMSE) похибки.

Наприклад, для найближча точка між еліпсом на прямою **u(x)** має координати (5.6 12.6):

$x := 2$

Given $\quad f(x,y) = 0 \quad y + x = 20 \quad z := \text{Minerr}(x,y) = \begin{pmatrix} 5.616 \\ 12.635 \end{pmatrix}$

$\text{ERR} = 1.857$

Застосування **minerr** можна рекомендувати для обходу випадків, обчислювальний блок **given-find** не знаходить коренів для якогось набору значень параметрів системи.

Наприклад, рівняння параболоїду може мати наступний вигляд

$$kx^2 + y^2 = 0.$$

Для всіх додатних значень коефіцієнта **k** параболоїд перетинає вісь OX. До значень k≈700000 блок **given-find** забезпечує знаходження коренів. для більших значень коефіцієнта **k** функція **find** не справляється з розв'язанням, а функція **minerr** знаходить відповідь:

$k := 10^6$ $x := 1$ $y := 1$

given $k \cdot x^2 + y^2 = 0$ find(x ,y) =

Given $k \cdot x^2 + y^2 = 0$ minerr(x ,y) = $\begin{pmatrix} -4.912 \times 10^{-4} \\ 7.091 \times 10^{-4} \end{pmatrix}$

Блок **given** може застосовуватися також для розв'язання одного рівняння. Слід пам'ятати, що на відміну від функції **root**, блок **given** визначає ДІЙСНИЙ корінь. Тому можуть трапитися випадки, коли при невдалому початковому наближенні блок **given** відповіді не дає.

Наприклад, для рівняння (3.10) застосування обчислювального блока з початковим наближенням -5 відповіді не дає:

$y(x) := 2 \cdot x^5 + 20 \cdot x^2 - 2 \cdot x + 100$

$z := -5$ given $y(z) = 0$ find(z) =

$z := -10$ Given $y(z) = 0$ Find(z) = −10.545

Символьні засоби розв'язання рівнянь та систем рівнянь MathCAD

СКМ MathCAD дозволяє символьними перетвореннями отримати розв'язання рівнянь, в тому числі параметричних, та систем рівнянь в аналітичному або в чисельному вигляді.

Запропонований пакетом символьний результат може мати неочікуваний вигляд. В таких випадках потребуються додаткові дії перетворення для надання результатам прийнятного вигляду.

Іноді символьна відповідь виражається через спеціальні функції, деякі з яких в подальшому можуть бути оброблені в пакеті тільки символьними засобами.

Найпростіший спосіб отримання символьної відповіді – це застосування оператора символьного виведення результату «→» замість оператора обчислення «=». Вводиться цей оператор кнопкою з символьної панелі, панелі обчислень або комбінацією клавіш **Ctrl+.**.

Серед функцій чисельного розв'язання рівнянь та систем рівнянь для символьного застосування найбільш пристосованим є блок **given-find**. Він може застосовуватися для символьного розв'язання будь яких рівнянь та систем рівнянь. Інші функції мають обмежуючі особливості для аналітичного застосування. Недоліком подібного способу є неможливість визначення умови, обмежень для розв'язання або призначити не одну, а кілька послідовних дій обробки даних.

Наприклад,

given

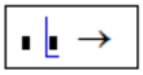
$$x^2 + y^2 = 1$$

$$\text{find}(y) \rightarrow \left(\sqrt{x-1}\cdot\sqrt{x+1}\cdot i \quad -\sqrt{x-1}\cdot\sqrt{x+1}\cdot i\right)$$

Більш універсальним є застосування символьного шаблона з ключем, який визначає дію для виконання.

Введення проводиться:

- кнопкою шаблона дії з ключовим словом з символьної панелі

  з наступним заповненням першого маркера вихідними даними, другого – написом ключового слова дії або введенням шаблона комбінацією клавіш **Ctrl+>** з наступним заповненням маркерів шаблона,
- натисканням клавіші відповідної дії символьної панелі з наступним введенням в поле маркера вихідних даних.

Оператори символьних перетворень можуть поєднуватися в ланцюг дій для послідовного виконання. Таким діями після застосування основного оператора дії є спрощення **simplify**, отримання чисельного результату **float**, висунення умови до типу результату **assume**.

Для введення додаткової ланки в ланцюжок слід після введення першого символьного оператора ввести комбінацію **Ctrl+Shift+.** або після натискання відповідної кнопки першого символьного оператора на символьній панелі натиснути наступну.

Для отримання значень вектора коефіцієнтів алгебраїчного рівняння з метою застосування можна використати

В парі з функцією **polyroots** для розв'язання алгебраїчних рівнянь, коли рівняння записано в неявному вигляді, може застосовуватися символьний оператор **coeffs.** Оператор визначає вектор коефіцієнтів рівняння. Наприклад

$$xr := \left(x^2 + x + 1\right)^2 + (x-1)^3 + (x+1)^4 \text{ coeffs} \rightarrow \begin{pmatrix} 1 \\ 9 \\ 6 \\ 7 \\ 2 \end{pmatrix} \qquad \text{polyroots}(xr) = \begin{pmatrix} -2.981 \\ -0.2 + 1.169i \\ -0.2 - 1.169i \\ -0.119 \end{pmatrix}$$

Для розв’язання рівнянь та систем рівнянь в символьному вигляді призначено блок символьного розв’язання

solve →

.

В поле маркера блока вводиться вираз функції рівняння або вектор виразів функцій системи рівнянь. Для рівнянь в нормальному вигляді прирівнювання до нуля не потрібно, тобто достатньо введення лівої частини рівняння. В інших випадках запис має містити знак «жирного» логічного прирівнювання та значення правої частини рівняння. За необхідності після ключового слова через кому позначаються імена змінних, відносно яких потрібно знайти розв’язок.

Наприклад,

$$x^3 + 11x^2 + 3x - 135 = 0 \text{ solve} \rightarrow \begin{pmatrix} -9 \\ -5 \\ 3 \end{pmatrix} \qquad x - a + 5 \text{ solve},x \rightarrow a - 5$$

$$\begin{pmatrix} x^2 + y^2 = 1 \\ y - x^2 = 1 \end{pmatrix} \text{solve},x,y \rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ \frac{1}{2} & \\ i \cdot 3 & -2 \\ \frac{1}{2} & \\ -i \cdot 3 & -2 \end{pmatrix}$$

Оператор **solve** для алгебраїчних рівнянь шостої та вищих степенів видає чисельне значення кореня, бо не розроблено методів аналітичного розв'язання для алгебраїчних рівнянь степенів вище за 5.

Для тригонометричних рівнянь оператор **solve** шукає один корінь.

Наприклад,

$$\sin(a \cdot x)\ \text{solve},x \rightarrow 0$$

Отримати всі корені можна за спробувати додаванням в ланцюжок дій модифікатора **fully**. Наприклад,

$$\sin(a \cdot x)\ \text{solve},x,\text{fully} \rightarrow \begin{cases} \_c1 & \text{if } a = 0 \wedge \_c1 \in \mathbb{C} \\ \frac{\pi \cdot \_n}{a} & \text{if } a \neq 0 \wedge \_n \in \mathbb{Z} \end{cases}$$

Оператор **solve** виводить розв'язок у вигляді вектора-стовпця.

Для отримання потрібного вигляду результату символьного розрахунку може знадобитися кілька варіантів дій.

Наприклад, розв'язання рівняння $\mathbf{x^2+y^2=1}$ відносно **y** має наступний вигляд:

$$y = \mp\sqrt{1 - x^2}.$$

Застосування оператора символьного розв'язання виглядає як

$$x^2 + y^2 = 1\ \text{solve},y \rightarrow \begin{pmatrix} \sqrt{x-1} \cdot \sqrt{x+1} \cdot i \\ -\sqrt{x-1} \cdot \sqrt{x+1} \cdot i \end{pmatrix}.$$

Оператор видає вірну відповідь, але запис є незручним із використанням уявної одиниці. Додавання в ланцюжок дії символьного спрощення **simplify** не змінює вигляд відповіді:

$$x^2 + y^2 = 1 \quad \left| \begin{matrix} \text{solve},y \\ \text{simplify} \end{matrix} \right. \rightarrow \begin{pmatrix} \sqrt{x-1}\cdot\sqrt{x+1}\cdot i \\ -\sqrt{x-1}\cdot\sqrt{x+1}\cdot i \end{pmatrix}$$

Прийнятного вигляду відповідь набуває тільки після додаткового обмеження типу відповіді дійсними значеннями:

$$x^2 + y^2 = 1 \quad \left| \begin{matrix} \text{solve},y,\text{real} \\ \text{simplify} \end{matrix} \right. \rightarrow \begin{pmatrix} \sqrt{1-x^2} & 0 \\ -\sqrt{1-x^2} & 0 \end{pmatrix}.$$

Наприклад, вигляд відповіді оператора символьного розв'язання для рівняння (3.10) є дуже громіздким та незручним. Створення ланцюжків з додатковими діями спрощення, типу результату вигляд відповіді не змінюють. Тільки вимога символьно-чисельної форми надає відповіді звичної форми:

$$2\cdot x^3 + 20\cdot x^2 - 2\cdot x + 100 \quad \left| \begin{matrix} \text{solve},x,\text{real} \\ \text{float} \end{matrix} \right. \rightarrow \begin{pmatrix} -10.544528449924184116 & 0 \\ 0.27226422496209205808 - 2.1604786330404814207i & 0 \\ 0.27226422496209205808 + 2.1604786330404814207i & 0 \end{pmatrix}$$

Рекомендується прості рівняння та системи розв’язувати символьно, більш складні – чисельно.

Засоби Matlab

Безпосередньо для розв’язання системи нелінійних рівнянь $F(\vec{x}) = 0$ призначено функцію **`fsolve`**. Розв’язання ведеться різновидами методу **найменших квадратів**.

В разі запису цільової функції оптимізації у вигляді системи нелінійних рівнянь, для знаходження коренів системи можуть застосовуватись функції **`fminsearch`** (**симплекс метод**), **`fminunc`** (**метод Ньютона, квазіньютонівський метод DFP**), **`lsqnonlin`** (**метод НК, Ньютона або Левенберга**) з додатку **`Optimization toolbox`**.

Синтаксис всіх наведених функцій має однаковий вигляд:

```
[x,<fval>,<exitflag>,<output>,<jacobian>]=
MLABFUN(fun,x0,<options>,...),
```

де **x** – вектор коренів системи, **fun** – функція опису системи (цільової функції), **x0** – вектор початкових значень. Точка початку пошуку. **options** – опціональна структура опису умов роботи функції **fsolve.**

Функція **fun** приймає аргумент – вектор значень x, повертає результат – вектор значень рівнянь системи. В разі застосування методів розв'язання з використанням аналітично визначеного Якобіану, повинна містити розрахунок останнього. Використання аналітично визначеного Якобіану проводиться записом опції в параметрі **options=optimset('Jacobian','on').** В такому випадку функція опису системи повинна видавати результат у вигляді вектора з двох елементів. Перший елемент – вектор значень рівнянь, другий – матриця значень Якобіану.

Функція **optimset**, яка викликана з іменем функції **MLABfun** Matlab в якості аргументу, виводить в командне вікно перелік можливих установок для алгоритмів, які реалізовані в функції.

Додатково до полів, які використовуються функцією **fzero**, для **fsolve** доступні наступні поля:

Algorithm – визначає метод розв'язання: **'trust-region-dogleg' (собачої ноги, погоні,** за замовчанням), **'trust-region-reflective' (модифікований ньютонівський з відбиттям), 'levenberg-marquardt' (модифікований ньютонівський – Левенберга-Маркуардта).** За замовчанням параметр останнього метода $\lambda = 0.01$. Для зміни, наприклад, на 0.5 потрібно визначити комірку **{'levenberg-marquardt',0.05}.** Тільки останній метод може бути застосований для систем з розміром не **NxN**.

DerivativeCheck – перевірка значень чисельних похідних: '**on**'/ **'off'** (за замовчанням).

Diagnostics – виведення службової інформації про поведінку рівняння: **'on'**/ **'off'** (за замовчанням).

DiffMaxChange – максимальне допустиме значення градієнта (**Inf** за замовчанням).

DiffMinChange – мінімальне допустиме за амплітудою значення градієнта (0 за замовчанням).

FinDiffRelStep – вектор кроку в чисельній похідній. Для центральної похідної **delta = v.*max(abs(x),TypicalX).** Для однобічної похідної **delta= v.*sign(x).*max(abs(x),TypicalX).**

FinDiffType – тип чисельної похідної: **'forward'** (однобічна вперед, за замовчанням), **'central'** (центральна).

Jacobian – спосіб обчислення матриці Якобі: **'on'** – аналітичне в функції користувача, **'off'** (чисельно, за замовчанням).

MaxFunEvals – максимальна кількість обчислень функції. За замовчанням 100*кількість змінних.

MaxIter – максимальна кількість ітерацій. За замовчанням 400.

PlotFcns – додатково доступні наступні графіки: @**optimplotfunccount** – кількість обчислень функції, @**optimplotresnorm** – норма (модуль) нев'язки, @**optimplotstepsize** – крок.

TolFun – порогове значення функції рівняння. За замовчанням 1e-6.

TypicalX – **Trust-region-dogleg** алгоритм присвоює це значення елементам головної діагоналі масштабуючої матриці. За замовчанням дорівняю 1.

fval – опціональний результат. Вектор значень рівнянь системи в точці **x**.

exitflag – опціональний результат. Код ознаки закінчення дії функції:

1 Значення градієнта менше точності;

2 Крок X менше точності;

3 Крок зміни цільової функції менше точності;

4 Зсув в визначеному напрямку менший за точність.

0 Досягнуто максимальну кількість ітерацій;

-1 Алгоритм завершено по значенню функції;

-2 Алгоритм намагається визначити точку, яка не є коренем.

-3 Радіус області замалий або параметр регуляції завеликий.

Для **ньютонівського метода з відбиттям** додатково доступні поля:

`JacobMult` – посилання на функцію множення якобіану.

`JacobPattern` – матриця розряження якобіану для чисельного диференціювання.

`MaxPCGIter` – максимальна кількість ітерацій спряжіння градієнтів.

`TolPCG` – порогове значення спряжіння градієнтів. За замовчанням 0.1.

`output` – опціональний результат. Структура з описом ходу розв'язання системи.

`output.algorithm` – використаний алгоритм;

`output.funcCount` – кількість оцінок функції;

`output.iterations` – кількість ітерацій.

`jacobian` – опціональний результат. Матриця Якобі перших похідних рівнянь системи.

Наприклад, розв'язання системи

$$2x_1 - x_2 - e^{-x_1} = 0$$
$$2x_2 - x_1 - e^{-x_2} = 0$$

з параметрами за замовчанням (метод – **`dogleg`**, точність по функції та кореню – 1.e-6) з початковою точкою (-5 5) може виглядати наступним чином:

```
function F=myfun11(x)
F=[2*x(1)-x(2)-exp(-x(1));-x(1)+2*x(2)-exp(-x(2))];
```

виклик:

```
[x,fval]=fsolve(@myfun, [-5; 5])
```

відповідь:

```
Equation solved.fsolve completed because the vector
of function values is near  zeroas measured by the
```

default value of the function tolerance, and the problem appears regular as measured by the gradient.

x = 0.5671 0.5671

fval = 1.0e-006 * -0.4059 -0.4059

Символьні засоби

Символьне розв'язання рівнянь СКМ можливо за умови встановлення додатку **Symbolic Math Toolbox (Mupad).**

Функція

S = solve(expr<,var,Name,Value>)

видає при можливості символьну відповідь та чисельну відповідь, якщо не може знайти аналітичний розв’язок.

Обов'язковим аргументом функції є **expr** – вираз символьного типу або рядок з виразом рівняння. Рівняння в нормальному вигляді не потребують повного запису зі знаком прирівнювання та нуля в правій частині. Рівняння системи записуються через кому.

Наприклад,

```
solve('x+1=0')   solve('x+1')
a=sym('x+1');  solve(a);
solve('x+y=10','x*y=2')
```

Результатом функції для рівнянь є змінна **S** символьного типу. Для систем рівнянь можна визначати результат однією змінною або вектором-рядком. Одна змінна є структурою з полями, в яких розташовуються символьні змінні коренів системи. Для вектора, кожен елемент вектора є змінною символьного типу коренів системи .

Наприклад, для розв'язання рівняння **s=solve('x-1+y=0')** змінна **s** приймає значення **1-y,** для розв'язання системи **s=solve('x+y=10','x*y=2')** **s** стає структурою з полями **s.x=(**$5+\sqrt{23}$ $5-\sqrt{23}$**)** та **s.y=(** $5+\sqrt{23}$ $5-\sqrt{23}$**),** для запису з відповіддю у вигляді вектора **[s a]=solve('x+y=10','x*y=2')** результатом є два вектори символьного типу **s=[** $5+\sqrt{23}$ $5-\sqrt{23}$**]** та **a=[** $5+\sqrt{23}$ $5-\sqrt{23}$**].**

За замовчанням змінною рівняння вважається змінна **x**, змінними систем рівнянь – **x, y**. В разі необхідності змінну, відносно якої потрібно знайти розв'язання, можна записати вручну за допомогою опційного аргументу **var** у вигляді імені змінної символьного типу з попереднім визначенням функціями **sym**, **syms**, **symvar** або текстовим рядком забраним в апострофи.

Наприклад,

```
>> solve('x+y+a')    ans = - a - y
>> solve('x+y+a','a')  ans =- x - y
```

Для тонкого налаштування процесу обчислення слугують опційні параметри **Name, Value** у вигляді забраних в апострофи текстових рядків послідовності пар імені властивості та значення властивості. Для налаштування доступні максимальний ступінь алгебраїчних рівнянь, для яких визначається символьна відповідь **MaxDegree,** ігнорування математичних обмежень наприклад, діапазону значень під квадратним коренем **IgnoreAnalyticConstraints**, ігнорування сумісності відповіді з первинним типом змінної **IgnoreProperties,** кількість відповідей **PrincipalValue,** виведення тільки дійсних відповідей **Real.**

MaxDegree – натуральне число (1 - 5). За замовчанням 3.

IgnoreAnalyticConstraints – логічне значення (**false/true**). За замовчанням **false.**

Наприклад, розв'язання логарифмічного рівняння проводиться без спрощення виразу

```
solve(log(x*y) - log(y) - 5, 'x')
ans = exp(log(y) + 5)/y
```

Вмикання ігнорування обмежень дозволяє отримати простішу відповідь:

```
solve(log(x*y)-log(y)- 5, x,...
'IgnoreAnalyticConstraints', true)
ans =exp(5)
```

IgnoreProperties – логічне значення (**false/true**). За замовчанням **false.**

Наприклад, для додатної змінної **x** функція **solve** видасть тільки один додатний корінь:

```
syms x positive;
solve(x^2 + 5*x - 6, x)
ans = 1
```

Вмикання ігнорування типу змінної дозволяє отримати обидва корені:

```
solve(x^2 + 5*x - 6, x, 'IgnoreProperties', true)
ans =   1  -6
```

PrincipalValue – логічне значення (**false/true**). За замовчанням **false.** Значення **True** призводить до виведення тільки одного значення.

Real – логічне значення (**false/true**). За замовчанням **false.**

В версіях до 8.X функція **solve** перед своїм виконанням викликає функцію **symvar**, яка визначає змінні в виразі. В версіях пізніше 8.X не рекомендується визначення функції у вигляді текстового рядка. Наприклад, **solve('x+1')**. Пропонується для визначення символьних змінних та виразів використовувати функцію **syms**, перетворення текстового рядка в символьний проводити функцією **str2sym**. Наприклад,

```
syms x; y=x-2; solve(y)
s=str2sym(x-2); solve(s)
```

Контрольні запитання

Які перевірки виконуються в якості критерію розв'язання СНР?

Що є умовою застосування методу простої ітерації для СНР?

В чому полягають відмінності методу Зейделя?

Що є умовою застосування методу дотичних для СНР?

Які засоби має для розв'язання СНР MathCAD?

За яким методом працює блоковий оператор MathCAD?

Що є обов'язковим для застосування блокового оператора MathCAD?

Який тип розв'язку СНР знаходить блоковий оператор MathCAD?

В чому полягає відмінність блокового оператора та функції **minerr?**

Як встановлюються додаткові режими для блокового оператора MathCAD?

Як записати послідовність символьних дій в MathCAD?

Для чого призначено символьний оператор **coeffs?**

Як отримати множину коренів в операторі **solve?**

Для чого призначено функцію **fsolve**?

Як визначити метод розв'язання СНР в функції **fsolve**?

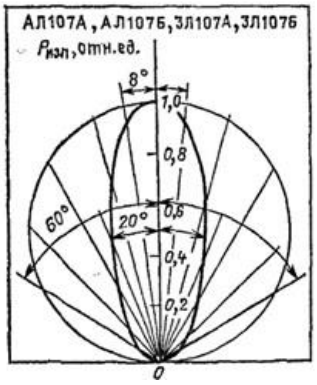
З якою точністю за замовчанням розв'язує СНР Matlab?

В чому полягає відмінність між функціями **solve** та **fsolve**?

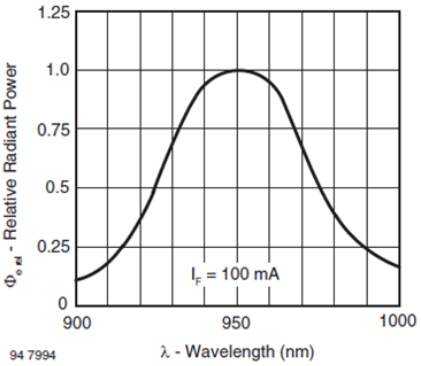
6. МЕТОДИ НАБЛИЖЕННЯ ФУНКЦІЙ

Задача наближення функцій виникає при розв'язанні багатьох практичних і теоретичних задач.

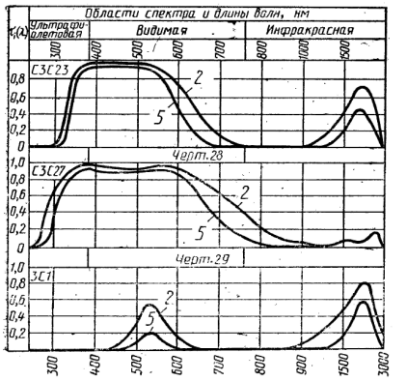
Часто вхідні дані визначені у вигляді таблиць чи графіків (рис. 6.1). У вигляді графіків зазвичай представлені спектральна та частотні характеристики елементів, індикатриси випромінювачів в документації виробників та довідниках, властивості матеріалів в ДСТУ тощо, у вигляді таблиць визначаються показник заломлення скла в ГОСТ, коефіцієнт ослаблення атмосфери за Пасмоном-Ларморм тощо.



а)



б)



в)

Длина волны монохроматического излучения λ, нм	V (λ)	Длина волны монохроматического излучения λ, нм	V (λ)
540	0,954	660	0,061
550	0,995	670	0,032
560	0,995	680	0,017
570	0,952	690	0,0082
580	0,870	700	0,0041
590	0,757	710	0,0021
600	0,631	720	0,00105
610	0,503	730	0,00052
620	0,381	740	0,00025
630	0,265	750	0,00012
640	0,175	760	0,00006
650	0,107		

г)

Рисунок 6.1 – Табличні та графічні дані: а – індикатриса випромінювача; б – спектральна характеристика світлодіода; в – пропускання скла; г – спектральна світлова ефективність

У вигляді масивів, що є аналогом таблиць, визначають експериментальні дані. Наприклад, графік **`f(x)`** експериментальних даних

здається схожим на якусь функціональну залежність, наприклад, параболу. Але за рахунок експериментальних похибок точки графіка не точно співпадають з очікуваною залежністю. Потрібно отримати найкраще в певному значенні наближення функції **f(x)** до очікуваної при мінімальній кількості вимірювань.

Крім того, бувають випадки, коли аналітичний вираз функції **f(x)** відомий, проте є занадто складним і незручним для обчислень, тому необхідно вихідну функцію замінити на більш просту. Наприклад,

$$e^x \approx \sum_{k=0}^{n} \frac{x^k}{k!},$$

$$\sin(x) \approx x + \sum_{k=1}^{n} (-1)^k \frac{x^{2k-1}}{(2k-1)!}.$$

Наближення функцій широко застосовується при розв'язанні інших обчислювальних задач комп'ютерного моделювання: чисельного інтегрування і диференціювання, розв'язання диференціальних рівнянь, розв'язання нелінійних рівнянь, задач оптимізації тощо.

Нехай є деяка функція (рис. 6.2) **f(x)**, яка в **N** точках $x_1, x_2, \ldots, x_N$ приймає, відповідно, значення $y_1, y_2, \ldots, y_N$, тобто $y_i = f(x_i)$, $i=1\ldots N$. Така функція називається **сітчастою** або **таблично визначеною**. Впорядкована сукупність точок $x_1, x_2, \ldots, x_N$. називається **сіткою**. В разі, коли відстань між точками аргументів є сталою, тобто $|x_i - x_{i+1}| = const$ – сітка називається **рівномірною**. Точки аргументів x_i називають **вузлами** сітки.

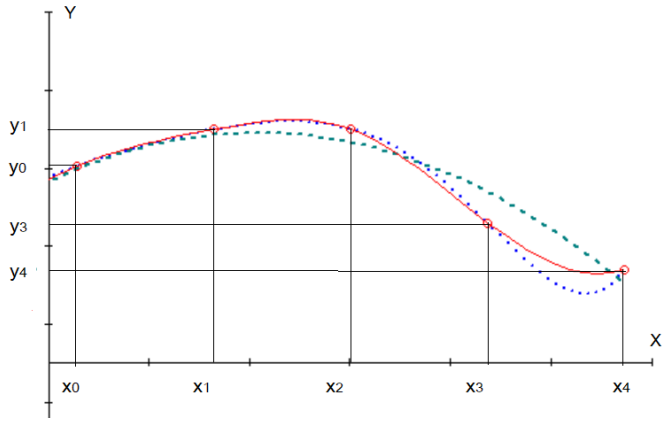


Рисунок 6.2 – Сітчаста функція

Задачею **наближення** є знаходження значень функції в точках **x**, які не співпадають з вузлами сітки, шляхом знаходження аналітичного виразу функції наближення **φ(x)**, яка в деякому сенсі близька до вихідної сітчастої функції **$f(x_i)$.**

Якщо для знаходження наближаючої функції використовуються значення первинної функції **$f(x_i)$** та сітки **$[x_1,\dots,x_N]$**, наближення є **глобальним**. Якщо використовуються тільки обрані значення в околі аргументу **x** – **локальним** або «**ковзаючим**». Різниця значень первинної функції та наближаючої функції називається **нев'язкою**.

Якщо в якості критерія наближення використовується умова співпадіння значень функції **φ(x)** зі значеннями даної функції

$$y_i=f(x_i)=\varphi(x_i),\ i=1\dots N,$$

наближення називають **інтерполяцією**. В інших точках з області визначення виконується наближена рівність **$f(x)\approx\varphi(x)$.** Функція **φ(x)** називається **інтерполюючою** функцією. З геометричної точки зору задача інтерполювання для функції однієї змінної означає побудову кривої, яка проходить через точки з координатами **$(x_1, y_1), \dots, (x_N, y_N)$.** Через дані точки можна провести безліч кривих. Тому задача інтерполювання в її загальному вигляді не є однозначно визначеною.

У випадках, коли функцію **f(x)** замінюють на функцію, близьку до **$f(x)\approx\varphi(x;a)$** в усьому діапазоні $x \in [x_1, x_N]$, де **a** – деякі параметри функції **φ**, процес наближення називають **апроксимацією**, а функцію **φ(x;a)** називають **апроксимуючою**.

Інтерполяція

Інтерполююча функція записується у вигляді

$$\varphi(x) = \sum_{i=-\infty}^{\infty} C_i\varphi_i(x). \qquad (6.1)$$

Якщо, наприклад, функція **f(x)** періодична, то функції **$\varphi_i(x)$** доцільно вибирати з класу тригонометричних функцій. У випадках, коли функція **f(x)** набуває безмежно великих значень в околі деякої точки, функції **$\varphi_i(x)$** вибирають з класу дробово-раціональних функцій.

Якщо функції $\varphi_i(x)$ вибирати з класу многочленів-поліномів

$$\varphi_i(x)=x^i,$$

то наближення називають **поліноміальним** або **параболічним** [17].

Поліноміальна інтерполяція (6.2) проста за формою, її легко обчислювати, диференціювати та інтегрувати.

$$\varphi(x) = P_N(x) = a_0x^0 + a_1x^1 + \cdots + a_Nx^N = \sum_{i=0}^{N} a_ix^i. \qquad (6.2)$$

$$P_N(x_i) = y_i = f(x_i),\ i = 0 \ldots N. \qquad (6.3)$$

Функціональна інтерполяція, коли в якості наближаючої функції використовується інша залежність (логарифмічна, показова, тощо), призводить до необхідності складання та розв'язання системи нелінійних рівнянь. Основним недоліком такого підходу є те, що система може не мати коренів. Кількість коефіцієнтів в функціональних залежностях не перевищує 2 - 3. Тому для інтерполяції доводиться обирати тільки кілька пар точок з усіх значень вузлів. Це призводить до залежності визначених коефіцієнтів від положення обраних вузлів та збільшенню похибки. Тому функціональна інтерполяція широкого вжитку не знаходить.

Поліноміальна глобальна інтерполяція

Для глобальної поліноміальної інтерполяції для сітки з N точок поліном завжди має степінь $M=N-1$.

Для визначення невідомих коефіцієнтів многочлена (6.2) побудуємо систему рівнянь на основі умов (6.3)

$$\begin{cases} a_0x_0^0 + a_1x_0^1 + \cdots + a_Nx_0^N = y_0 \\ a_0x_1^0 + a_1x_1^1 + \cdots + a_Nx_1^N = y_1 \\ \vdots \\ a_0x_N^0 + a_1x_N^1 + \cdots + a_Nx_N^N = y_N \end{cases},$$

$$\begin{bmatrix} x_0^0 & x_0^1 & \ldots & x_0^N \\ x_1^0 & x_1^1 & \ldots & x_1^N \\ \cdot & \cdot & \cdot & \cdot \\ x_N^0 & x_N^1 & \ldots & x_N^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix}. \qquad (6.4)$$

Вираз (6.4) є система $N+1$ лінійних рівнянь з $N+1$ невідомими відносно коефіцієнтів $a_0, a_i, \ldots, a_N$.

Система (6.4) має єдиний розв'язок, оскільки її визначник є визначником Вандермонда. Останній відмінний від нуля, бо за умовою задачі всі вузли інтерполювання різні. Таким чином, існує єдиний многочлен виду (6.2), що задовольняє умови (6.3) з коефіцієнтами

$$\vec{a} = X^{-1}\vec{y}.$$

Для зменшення кількості обчислень, уникнення необхідності розв'язання СЛАР для знаходження коефіцієнтів інтерполюючого полінома застосовуються **інтерполяційні формули Лагранжа** та **Ньютона**.

6.1 Інтерполяційний многочлен Лагранжа

Функція поліноміальної **глобальної інтерполяції за Лагранжем** для сітки з **N+1** точки [$x_0, \ldots, x_N$] є поліномом степеню **N** та шукається у вигляді

$$\varphi(x) = L_N(x) = C_0 l_0(x) + C_1 l_1(x) + \cdots + C_N l_N(x) = \sum_{i=0}^{N} C_i l_i(x),$$

де **Ci** – коефіцієнт, **li(x)** – функція Лагранжа.

Функція Лагранжа є поліномом степеню у вигляді добутків у вигляді різниці між значенням точки x та вузлів сітки xi за виключенням множника **$(x-x_i)$:**

$$l_i(x) = \alpha(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_N) = \alpha \prod_{k=0, k\neq i}^{N} (x - x_k).$$

І-та функція Лагранжа для k-го вузла сітки приймає наступні значення:

$$l_i(x_k) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}.$$

З урахуванням вищенаведеного і-та функція в і-му вузлі дорівнює 1:

$$l_i(x_i) = 1 = \alpha(x_i - x_0)(x_i - x_1) \ldots (x_i - x_N).$$

Звідки коефіцієнт α знаходиться як

$$\alpha = \frac{1}{\prod_{k=0, k\neq i}^{N} (x_i - x_k)}.$$

У вузлах сітки інтерполююча функція Лагранжа дорівнює значенням сітчастої функції **yi**:

$$L_N(x_i) = y_i = \sum_{k=0}^{N} C_k l_k(x_i) = C_i l_i(x_i) = C_i,$$

тобто $$C_i = y_i.$$

В результаті формула Лагранжа набуває наступного кінцевого вигляду:

$$L_N(x) = \sum_{i=0}^{N} \frac{y_i \prod_{k=0,k\neq i}^{N}(x-x_k)}{\prod_{k=0,k\neq i}^{N}(x_i-x_k)}. \quad (6.5)$$

Для того, щоб в циклах не проводити операцію перевірки введемо функцію

$$\omega_{N+1}(x) = \prod_{k=0}^{N}(x - x_k) \quad (6.6)$$

та її похідну у вузлі **xi**

$$\omega'_{N+1}(x_i) = \prod\prod_{k=0,i\neq k}^{N}(x_i - x_k).$$

Формула Лагранжа із застосуванням функції ω(x) набуває наступного вигляду

$$L_N(x) = \sum_{i=0}^{N} \frac{y_i \omega_{N+1}(x)}{(x-x_i)\omega'_{N+1}(x_i)}. \quad (6.7)$$

Формула Лагранжа є ефективною коли потрібно на незмінній сітці розраховувати значення кількох функцій.

6.2. Інтерполяційні формули Ньютона

Наближаюча функція **глобальної поліноміальної інтерполяції** за **Ньютоном** для рівномірної сітки з **N+1** точок записується у вигляді

$$\varphi(x) = N_N(x) = C_0 + C_1(x - x_0) + C_2(x - x_0)(x - x_1) + \cdots$$

$$+C_N(x - x_0)(x - x_1) \dots (x - x_{N-1}).$$

Вираз описує многочлен степеню **N** (степінь многочлену на одиницю менша за кількість вузлів сітки).

Скінченою різницею першого порядку називають різницю між значеннями функції у сусідніх вузлах інтерполювання:

$$\Delta^{(1)} y_i = y_{i+1} - y_i, \quad i = 0 \dots N - 1.$$

Скінчені різниці другого порядку утворюють зі скінченних різниць першого порядку:

$$\Delta^{(2)} y_i = \Delta^{(1)} y_{i+1} - \Delta^{(1)} y_i, \quad i = 0 \dots N-2.$$

Відповідно, **скінчена різниця** порядку **k** – це різниця скінчених різниць порядку **k-1** значень функцій в сусідніх точках для всіх **2<k<N**:

$$\Delta^{(k)} y_i = \Delta^{(k-1)} y_{i+1} - \Delta^{(k-1)} y_i, \quad i = 0 \dots N-k.$$

Скінчених різниць порядку **k** є рівно **N-k+1**, тобто із зростанням порядку скінчених різниць їх кількість зменшується.

Максимальний порядок скінченної різниці для сітки з **N** вузлів є **k=N-1**.

Наприклад, розрахуємо скінчені різниці для функції **$y=x^3-4x^2+7x-3$** для сітки **x=[0 … 4]** з кроком **h=1** [17].

Сітка має 5 вузлів, відповідно максимальний порядок скінчених різниць – 4.

x	y	Δ(1)y	Δ(2)y	Δ(3)y	Δ(4)y
0	-3	Δy0=y1-y0=4	Δ(2)y0=Δ(1)y1-Δ(1)y0=-2	Δ(3)y0=Δ(2)y1-Δ(2)y0=6	Δ(4)y0=Δ(3)y1-Δ(3)y0=0
1	1	Δy1=y2-y1=2	Δ(2)y1=Δ(1)y2-Δ(1)y1=4	Δ(3)y1=Δ(2)y2-Δ(2)y1=6	
2	3	Δy2=y3-y2=6	Δ(2)y2=Δ(1)y3-Δ(1)y2=10		
3	9	Δy3=y4-y3=16			
4	25				

Розглянемо інтерполяційний поліном Ньютона для рівномірної сітки з кроком **$x_{k+1}-x_k=h$**.

$$N_N(x_0) = C_0 = y_0,$$

$$N_N(x_1) = C_0 + C_1(x_1 - x_0) = y_0 + C_1 h = y_1,$$

$$C_1 = \frac{y_1 - y_0}{h} = \frac{\Delta^{(1)} y_0}{h},$$

$$N_N(x_2) = C_0 + C_1(x_2 - x_0) + C_2(x_2 - x_0)(x_2 - x_1) = C_0 + C_1 2h + C_2 2h^2 = y_2$$

$$C_2 = \frac{y_2 - C_0 - C_1 2h}{2h^2} = \frac{y_2 - y_0 - 2h\dfrac{y_1 - y_0}{h}}{2h^2} = \frac{(y_2 - y_1) - (y_1 - y_0)}{2h^2}$$
$$= \frac{\Delta^{(1)} y_1 - \Delta^{(1)} y_0}{2h^2} = \frac{\Delta^{(2)} y_0}{2h^2},$$

$$N_N(x_3) = C_0 + C_1(x_3 - x_0) + C_2(x_3 - x_0)(x_3 - x_1)$$
$$+ C_3(x_3 - x_0)(x_3 - x_1)(x_3 - x_2) = C_0 + C_1 3h + C_2 6h^2 + C_3 6h^3$$
$$= y_3,$$
$$C_3 = \frac{y_2 - C_0 - C_1 3h - C_2 6h^2}{6h^3} = \frac{\Delta^{(2)} y_1 - \Delta^{(2)} y_0}{6h^3} = \frac{\Delta^{(3)} y_0}{6h^3}.$$

В загальному випадку

$$C_k = \frac{\Delta^{(k)} y_0}{(k!)h^k}.$$

З урахування $\mathbf{C_k}$ формула Ньютона має наступний вигляд

$$\varphi(x) = N_N(x) = \ C_0 + \sum_{i=1}^{N-1} C_i \prod_{k=1}^{i}(x - x_{k-1}) =$$

$$= y_0 + \sum_{i=1}^{N-1} \frac{\Delta^{(i)} y_0}{i! h^i} \prod_{k=1}^{i}(x - x_{k-1}). \qquad (6.8)$$

Формулу Ньютона (6.8) застосовують коли необхідно провести обчислення сітчастої функції при зміні вузлів сітки, тобто провести «згущення» сітки.

На практиці використовують **формулу Ньютона** з аргументом у вигляді відносної координати – дійсного числа **t**.

В разі, коли **t** є додатним дійсним числом **0<t<N, x=x0+th**:

$x - x_0 = th$,

$x - x_1 = x - (x_0 + h) = x - x_0 - h = th - h = h(t-1)$,

$x - x_2 = x - (x_0 + 2h) = x - x_0 - 2h = th - 2h = h(t-2)$,

$x - x_i = x - (x_0 + ih) = h(t-i)$.

$$N_N(x) = N_N(x_0 + ht) = y_0 + \frac{\Delta^{(1)} y_0 (x - x_0)}{h} + \frac{\Delta^{(2)} y_0 (x - x_0)(x - x_1)}{2h^2} + \cdots +$$

$$\frac{\Delta^{(i)} y_0}{i! h^i} \prod_{k=1}^{i} (x - x_{k-1}) = y_0 + \frac{\Delta^{(1)} y_0 th}{h} + \frac{\Delta^{(2)} y_0 h^2 t(t-1)}{2h^2} + \frac{\Delta^{(i)} y_0 h^i}{i! h^i} \prod_{k=0}^{i-1} (t - k) =$$

$$y_0 + \sum_{i=1}^{N} \frac{\Delta^{(i)} y_0 \prod_{k=0}^{i-1} (t-k)}{i!}. \qquad (6.9)$$

Рівність (6.9) називають **першою інтерполяційною формулою Ньютона**. Цю формулу зручно використовувати для інтерполювання на початку відрізку інтерполяції. Кажуть, що першу інтерполяційну формулу Ньютона застосовують для **інтерполювання «вперед»**.

Коли значення аргументу знаходиться ближче до кінця таблиці скінченних різниць користуються **другою інтерполяційною формулою Ньютона**.

Базовою точкою відліку вважається останній вузол сітки $\mathbf{x_N}$.

В цьому випадку **t** є від'ємним дійсним числом **0>t>-N, $\mathbf{x=x_N+th}$**.

$\mathbf{x-x_N=th}$,

$\mathbf{x-x_N-1=x-(x_N-h)=x-x_N+h=th+h=h(t+1)}$,

$\mathbf{x-x_N-2=x-(x_N-2h)=x-x_N+2h=th+2h=h(t+2)}$,

$\mathbf{x-x_N-i=x-(x_N-ih)=h(t+i)}$.

Функція Ньютона набуває наступного вигляду:

$$\varphi(x) = N_N(x) = C_0 + C_1(x - x_N) + C_2(x - x_N)(x - x_{N-1}) + \cdots$$

$$+C_N(x - x_N)(x - x_{N-1}) \ldots (x - x_1).$$

Інтерполяційний поліном запишеться як

$$N_N(x) = N_N(x_0 + ht) = y_{\mathrm{N}} + C_1(x - x_{\mathrm{N}}) + C_2(x - x_{\mathrm{N}})(x - x_{\mathrm{N-1}}) + \cdots$$

$$C_0 = y_N, \quad C_i = \frac{\Delta^{(i)} y_{N-k}}{i!\, h^i},$$

$$N_N(x) = N_N(x_0 + ht) = y_{\mathrm{N}} + \mathrm{t}\Delta^{(1)} y_{N-1} + \frac{\Delta^{(2)} y_{N-2}}{2} \mathrm{t(t+1)} + \cdots$$

$$+ \frac{\Delta^{(N)} y_0}{N!} \mathrm{t(t+1)} \ldots \mathrm{(t+N-1)} = y_{\mathrm{N}} + \sum_{i=1}^{N} \frac{\Delta^{(i)} y_{\mathrm{N-i}} \prod_{k=0}^{i-1} (t+k)}{i!}. \qquad (6.10)$$

Рівність (6.10) називають **другою інтерполяційною формулою Ньютона**. Цю формулу зручно використовувати для тих значень аргументу, що лежать в кінці відрізку інтерполяції. Кажуть, що другу інтерполяційну формулу Ньютона застосовують для **інтерполювання «назад»**.

Похибка глобальної поліноміальної інтерполяції

Оцінимо похибку інтерполяційної формули $R_N(x)$, тобто визначимо наскільки близько побудований многочлен наближається до функції $f(x)$ в точках, відмінних від вузлів сітки.

Формули Лагранжа та Ньютона є математично тотожними, тому

$$R_N(x)=f(x)-L_N(x)=f(x)-N_N(x).$$

Нехай функція $f(x)$ в має всі похідні до порядку N+1 включно.

Розглянемо функцію $u(x)=f(x)-LN(x)-k\omega_N(x)$,

де k – сталий множник, $\omega_N(x)$ – поліном (6.3).

Функція $\omega(x)$ має $N+1$ корінь в точках $x_0, x_i,\dots,x_N$. Функція $u(x)$ має ($N+2$)-й корінь у довільній фіксованій точці $x\in[x_0, x_N]$. В цій точці

$$u(x)=f(x)-L_N(x)-k\omega_{N+1}(x)=0.$$

Звідки визначаємо

$$k=\frac{f(x)-L_N(x)}{\omega_{N+1}(x)}=\frac{R_N(x)}{\omega_{N+1}(x)},$$

$$R_N(x)=k\omega_{N+1}(x).$$

При цьому значенні k функція $u(x)$ має ($N+2$)-й корінь і буде перетворюватися в нуль на кінцях відрізків $[x_0,x_1], [x_1,x_2],\dots, [x_i,x], [x,x_{i+1}],\dots, [x_{N-1},x_N]$.

Розглянемо ($N+1$) похідну функції $u(x)$.

За теоремою Ролля $u^{(N+1)}(x)=0$.

$$u^{(N+1)}(x)=f^{(N+1)}(x)-k(N+1)=0.$$

Звідки

$$k=\frac{f^{(N+1)}(x)}{(N+1)!}.$$

Підставляючи значення **k** в вираз для похибки $\mathbf{R_N(x)}$ остаточно отримуємо

$$\Delta_N \leq |R_N(x)| = \frac{|f^{(N+1)}(x)|}{(N+1)!}\omega_{N+1}(x). \qquad (6.11)$$

Абсолютна гранична похибка глобальної поліноміальної інтерполяції по **N+1** точках (степеню **N**) пропорційна значенню (**N+1**)-ї похідної вихідної сітчастої функції, поліному вузлів сітки $\boldsymbol{\omega_{N+1}(x)}$ та зворотно пропорційна факторіалу кількості вузлів сітки.

Для формул Ньютона з аргументом у вигляді відносної координати **t** похибка оцінюється як

$$|R_N(x)| = \frac{|f^{(N+1)}(x)|}{(N+1)!}h^{N+1}\prod_{i=0}^{N}(t-i). \qquad (6.11')$$

Слід зауважити, що між вузлами сітки інтерполяційні поліноми, як правило, нестійкі до погрішностей обчислень, причому нестійкість зростає зі збільшенням кількості вузлів сітки. Крім того, навіть невеликі погрішності значень **yi** можуть сильно змінити поведінку полінома між вузлами. Зазначені ефекти нестійкості виявляються вже при N > 15 [18].

6.3. Поліноміальна «ковзаюча» інтерполяція

Для сітчастих функцій з великою кількістю вузлів використання глобальної інтерполяції потребує невиправданих обчислювальних витрат, а похибка глобальної інтерполяції може перевищувати допустимі значення.

В таких випадках доцільним є застосування «**ковзаючої**» інтерполяції, коли до уваги беруться тільки значення сітчастої функції в околі точки x.

Найбільш поширеними є випадки лінійного і квадратичного інтерполювання. Відповідні формули можна отримати із загальної формули інтерполяції Лагранжа для **N = 1** і **N = 2**.

Для **N=1**, коли значення функції **f(x)** задано в двох вузлах $\mathbf{x_0}$ і $\mathbf{x_1}$, отримаємо формулу **лінійного інтерполювання**:

$$L_1(x) = \frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1.$$

Інтерполяційний многочлен є многочленом першого степеня, який є рівнянням прямої, що проходить через точки ($\mathbf{x_0, y_0}$) і ($\mathbf{x_1, y_1}$).

Для **N = 2** дістанемо формулу квадратичного інтерполювання:

$$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2.$$

Інтерполяційний поліном є квадратичною функцією, графіком якої є парабола, що проходить через три визначені точки (**x_0,y_0**), (**x_1,y_1**), (**x_2,y_2**).

Кускова лінійна інтерполяція

Кускова-лінійна інтерполяція полягає в тому, що на кожному окремому інтервалі [**x_i,x_{i+1}**] функція **f(x)** замінюється лінійною функцією (рис. 6.3)

$$\varphi_i(x) = \frac{x-x_{i+1}}{x_i-x_{i+1}}y_i + \frac{x-x_i}{x_{i+1}-x_i}y_{i+1}. \qquad (6.12)$$

Інтерполююча функція є ламаною, наче "зшитою" з відрізків прямих, кожен з яких розглядається на своєму "куску" .

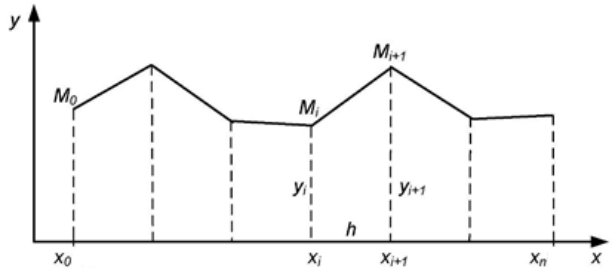


Рисунок 6.3 – Кускова лінійна інтерполяція

Схема кускової лінійної інтерполяції складається з двох кроків: знаходження інтервалу [**x_i,x_{i+1}**], якому належить точка x на першому кроці, та обчислення наближеного значення функції **f(x)** у точці x за формулою кусково-лінійної інтерполяції (6.12) – на другому.

Кускова квадратична інтерполяція

Подібно до кускова лінійної інтерполяції для наближення на кожному окремому інтервалі [**x_i,x_{i+1}**] функція **f(x)** наближається квадратичною (параболічною) функцією (рис. 6.4). Інтерполююча функція є «зшитою» з парабол, кожна з яких розглядається на своєму «кусочку» .

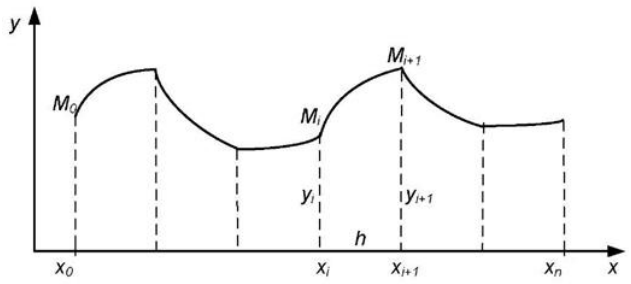


Рисунок 6.4 – Кусково-квадратична інтерполяція

Схема кускової квадратичної інтерполяції складається з двох кроків:

1. Знаходження номеру інтервалу $[\mathbf{x_i, x_{i+1}}]$, якому належить точка **x**.
2. В разі знаходження точки **x** в першому інтервалі, розрахунок наближення з використанням точок $\mathbf{a=x_i,\ b=x_{i+1},\ c=x_{i+2}}$, в разі знаходження в останньому інтервалі – з використанням точок $\mathbf{a=x_{i-2},\ b=x_{i-1},\ c=x_i}$, в інших випадках – із використанням точок $\mathbf{a=x_{i-1},\ b=x_i,\ c=x_{i+1}}$ за формулою кусково-квадратичної інтерполяції (3.13)

$$\varphi_i(x) = \frac{(x-x_b)(x-x_c)}{(x_a-x_b)(x_a-x_c)} y_a + \frac{(x-x_a)(x-x_c)}{(x_b-x_a)(x_b-x_c)} y_b + \frac{(x-x_a)(x-x_b)}{(x_c-x_a)(x_c-x_b)} y_c. \qquad (6.13)$$

Сплайн-інтерполяція

Кускова лінійна і **кускова квадратична** інтерполяція не дозволяють добитися гарного наближення, бо мають злами у вузлах сітки та велику похибку між вузлами. Одним із способів добитися кращого наближення є інтерполяція за допомогою **сплайн-функцій**, яка називається **сплайн-інтерполяцією**.

Сплайном (англ. **Spline** – рейка, лінійка) називається **кускова поліноміальна** функція, визначена на відрізку $[\mathbf{x_0, x_N}]$ і яка має на цьому відрізку не менше двох неперервних похідних:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + e_i(x - x_i)^4 + \cdots$$

$$i = 1 \ldots N. \qquad (6.14)$$

Сплайн - це функція, яка на кожному частинному відрізку інтерполяції є алгебраїчним многочленом (6.14) та є неперервною з кількома своїми похідними на всьому заданому відрізку.

На практиці знайшли застосування сплайни **лінійні** – **lspline**, **квадратичні** – **psline**, **кубічні** – **cspline**.

Збільшена точність **сплайн-інтерполяції** забезпечується додатковими умовами. Крім умови інтерполяції

$$\varphi_i(x_i) = y_i.$$

вимагається виконання умов рівності значень першої та другої похідних в спільних вузлах для суміжних «кусків» – підінтервалів інтерполяції:

$$\varphi_i'(x_i) = \varphi_{i+1}'(x_i) \quad \varphi_i''(x_i) = \varphi_{i+1}''(x_i). \tag{6.15}$$

та **граничних** умов: визначених значень першої та другої похідної функції **φ(x)** на межах інтервалу [**x0, xN**]. Якщо значення перших похідних на межах невідомі, то визначають рівною нулю другу похідну

$$\varphi_i''(x_0) = \varphi_i''(x_N) = 0.$$

Розглянемо кубічну сплайн-інтерполяцію, коли на кожному інтервалі **[x_{i-1},x_i], i=1,…,N** функція **S_i(x)** задається поліномом третього степеня:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3.$$

Для того щоб побудувати кубічний сплайн **S(x)** необхідно знайти всі коефіцієнти **a_i, b_i, c_i, d_i** поліномів **S_i(x), i=1,…,N**. Кількість цих невідомих коефіцієнтів дорівнює **4N**.

З умов інтерполяції:

справа: $S_i(x_i) = y_i = a_i + b_i(x_i - x_i) + \cdots = a_i,\ i = 1, N,$

зліва:$S_i(x_{i-1}) = y_{i-1} = a_i + b_i(x_{i-1} - x_i) + \mathrm{c}_i(x_{i-1} - x_i)^2 + d_i(x_{i-1} - x_i)^3 = y_i - b_i h + \mathrm{c}_i h^2 - d_i h^3,\ i = 1, N.$

З умов рівності значень перших похідних отримуємо два додаткові рівняння

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2,$$

$$S_i'(x_i) = b_i = S_{i+1}'(x_i) = b_{i+1} + 2c_{i+1}h + 3d_{i+1}h^2,\ \ i = 1, N - 1.$$

З умов рівності значень других похідних отримуємо останнє рівняння

$$S_i''(x) = 2c_i + 6d_i(x - x_i),$$

$$S_i''(x_i) = 2c_i = S_{i+1}'(x_i) = 2c_{i+1} + 6d_{i+1}h,\ \ i = 1, N - 1.$$

Система рівнянь для знаходження коефіцієнтів **a_i, b_i, c_i, d_i** є СЛАР наступного вигляду

$$\begin{cases} y_{i+1} = y_i + b_i h + \mathrm{c}_i h^2 + d_i h^3, & i = 1, N \\ b_i = b_{i+1} + 2c_{i+1} h + 3d_{i+1} h^2, & i = 1, N-1. \\ c_i = c_{i+1} + 3d_{i+1} h, & i = 1, N-1 \end{cases}$$

Система дозволяє визначити **3N - 2** лінійних рівняння для пошуку **3N** невідомих. Тому необхідно довизначити ще два рівняння з граничних умов:

$$S_N{}''(x_N) = c_N + 3d_N(x_N - x_N) = c_N = 0,$$
$$S_1{}''(x_0) = c_1 + 3d_1(x_0 - x_1) = c_1 - 3d_1 h = 0.$$

Таким чином, для обчислення коефіцієнтів сплайнів необхідно розв'язати СЛАР:

$$\begin{cases} y_{i+1} = y_i + b_i h + \mathrm{c}_i h^2 + d_i h^3, & i = 1, N \\ b_i = b_{i+1} + 2c_{i+1} h + 3d_{i+1} h^2, & i = 1, N-1. \\ c_i = c_{i+1} + 3d_{i+1} h, & i = 1, N-1 \\ c_3 = 0 \\ c_1 - 3d_1 h = 0 \end{cases}$$

Перевагою **сплайн-інтерполяції** є її стійкість до процесу обчислень і достатньо висока точність.

Схема сплайн-інтерполяції

1. Один раз визначити всі коефіцієнти розв'язанням СЛАР.
2. Знайти інтервал [$\mathbf{x_i, x_{i+1}}$], якому точка x належить.
3. Обчислити наближене значення функції **f(x)** в точці **x** як $\mathbf{S_i(x)}$ за виразом (6.14).

6.4. Засоби інтерполяції СКМ MathCAD, Matlab

Знаходження коефіцієнтів $\mathbf{a_i}$ інтерполюючого поліному зводиться до розв'язання СЛАР:

$$X\vec{a} = \vec{y}\ ,$$

де **X** – матриця Вандермонде значень координат вузлів сітки, **y** – вектор значень сітчастої функції, **a** – вектор шуканих коефіцієнтів полінома.

Пряме розв'язання завдання інтерполяції має наступний алгоритм:

1. Визначення векторів аргументу «x» та табличної функції «y» в N вузлах $\mathbf{x=(x_0, x_1 ... x_{N-1})}$ $\mathbf{y=(y_0, y_1 ... y_{N-1})}$.

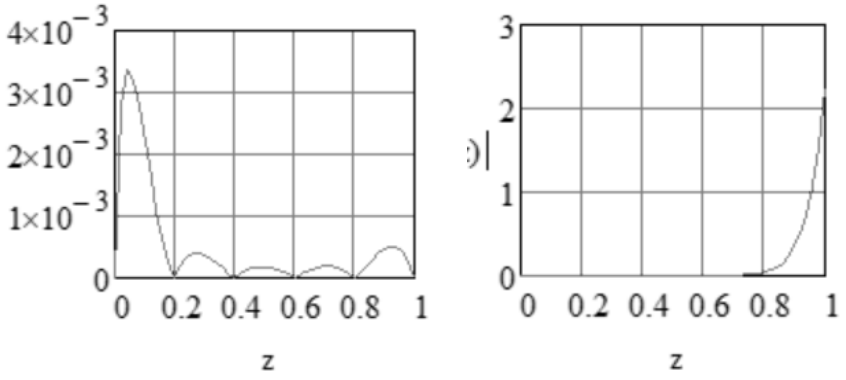
2. Визначення матриці коефіцієнтів рівнянь системи через вектор аргументів **x**.
3. Знаходження вектора коефіцієнтів **a** розв'язанням СЛАР.

Розрахунок значення інтерполяційного полінома в визначеній точці **z** за виразом (6.2).

В MathCAD розв'язання СЛАР поліноміальної інтерполяції можливо матричним методо $\vec{a} = X^{-1}\vec{y}$ та функцією **lsolve**.

В випадках, коли сітчаста функція визначена на великій кількості точок, інтерполяційний поліном стає занадто громіздким. Порядок СЛАР глобальної інтерполяція, похибки та час розв'язання стають завеликими навіть для сучасних СКМ. Крім того, зі зростанням ступеню поліному збільшується кількість його екстремумів, що погіршує точність інтерполяції між вузлами сітки.

Наприклад [12], для функції $y(x) = \sin(x)e^{-2\sqrt{x}}$ в діапазоні x від 0 до 1 в 6-ти точках абсолютна похибка інтерполяції з прямим розв'язанням СЛАР в матричному вигляді з обертанням матриці становить ~0.003 (рис. 6.5а), а для 26-ти точок – вже ~2 (рис. 6.5б)



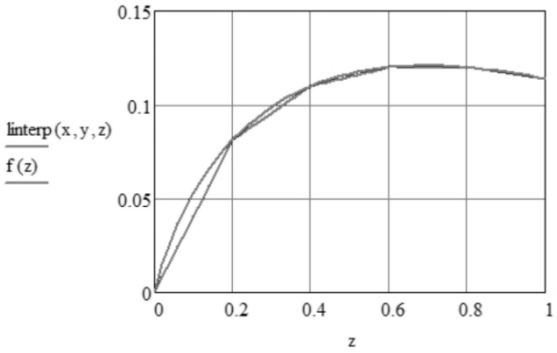
а) 6 вузлів б) 26 вузлів

Рисунок 6.5 – Похибки поліноміальної інтерполяції

Найпростіша кускова лінійна інтерполяція в СКМ MathCAD проводиться функцією **linterp(vx,vy,x)**, яка розраховує лінійно наближене значення табличної функції зі значеннями з аргументів векторів **vx,vy** в точках аргументу **x**. Для аргументу **x** у вигляді числа відповідь є числом, для аргументу **x** у вигляді вектора відповідь є вектором.

Наприклад, результат кускової лінійної інтерполяції функцією **linterp** для вищенаведеної функції $y(x) = \sin(x)e^{-2\sqrt{x}}$ може мати вигляд, який показано на рис. 6.6.

Зверніть увагу, що «середньоквадратична похибка для лінійної кускової інтерполяції складає 0.0044, а для глобальної інтерполяції 0.00098. Відносні похибки складають 3.6 % та 0.8 %, відповідно» [12].



$$\text{norm}(a) := \sqrt{\sum_{i=0}^{\text{last}(a)} \frac{(a_i)^2}{\text{length}(a)}}$$

$$lf := \text{linterp}(x, y, z)$$

$$\text{norm}(lf - f(z)) = 4.397 \times 10^{-3}$$

$$\text{norm}(fi(a, z, N) - f(z)) = 9.821 \times 10^{-4}$$

$$\frac{\text{norm}(lf - f(z))}{\text{norm}(fi(a, z, N) - f(z))} = 4.477$$

$$\frac{\text{norm}(lf - f(z))}{\max(f(z))} = 0.036 \qquad \frac{\text{norm}(fi(a, z, N) - f(z))}{\max(f(z))} = 0.008$$

Рисунок 6.6 – Лінійна кускова інтерполяція

Базовою функцією СКМ MathCAD для розрахунку наближення сітчастої функції **vx, vy** в визначеній точці **x** є функція

interp(vs,vx,vy,x) ,

де **vs** – допоміжний вектор коефіцієнтів, **vx, vy** – вектори/матриці аргументів та значень табличної функції відповідно. Для двомірних функції **vx** є матрицею розміром **Nx2**, елементами якої є координати точок діагоналі сітки, **vy** є матрицею розміром **NxN** зі значеннями сітчастої сітки в вузлах, **x** – точка/ вектор, в якій необхідно обрахувати значення.

Функція **interp** проводить наближення функції методами **сплайн** інтерполяції та **поліноміальної** інтерполяції/апроксимації. Метод розрахунку визначається способом підготовки допоміжного вектора **vs.**

Для сплайн інтерполяції допоміжний вектор **vs** розраховується наступними вбудованими функціями:

Функція **vs=lspline(vx,vy)** готує коефіцієнти лінійного сплайна,

Функція **vs=pspline(vx,vy)** - квадратичного сплайна,

Функція **vs=cspline(vx, vy)** – кубічного сплайна.

Функція **vs=bspline(vx, vy, u, n)** є універсальною. Параметр **n** визначає: 1 – лінійний сплайн, 2 – квадратичний, 3 – кубічний. Функція розраховує допоміжний вектор коефіцієнтів **vs** в векторі **u** довжиною **N-1** таким чином, що стикування сплайнів проводиться не в вузлах сітки, а в точках, які визначає користувач. Розмірність вектора u повинна бути на 1 для лінійного сплайна (2 для квадратичного, 3 для кубічного) меншою за розмірність векторів **vx, vy.** Значення точок повинні відповідати наступним умовам **u1≥vx1, uN≤vxN**.

Слід відмітити, що сплайн інтерполяція точніша за звичайну «ковзаючи» інтерполяцію, але не обов'язково для будь-яких випадків точніша за глобальну поліноміальну інтерполяцію.

Наприклад, на рис. 6.7 наведено похибки для 26-ти точок наведеного вище прикладу. З графіків видно, що найменшу похибку має глобальна інтерполяція. Далі в порядку зменшення точності йдуть кубічна, параболічна, лінійна сплайн інтерполяції. Найгірші результати має кускова лінійна інтерполяція [12].

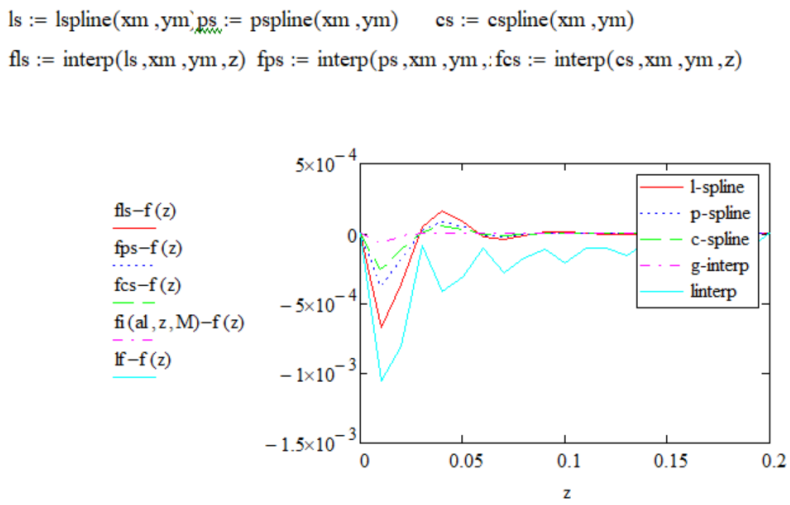


Рисунок 6.7 – Похибки сплайн інтерполяції

Засоби Matlab

В СКМ Matlab розв'язання СЛАР інтерполяції може проводитися матричним методом ($\vec{a} = X/\vec{x}$). Розрахунок матриці Вандермонде **X** з вектора **x** ефективніше провести призначеною для цього вбудованою функцією **vander(x).**

Вбудовані функції інтерполяції є зовнішніми m-функціями Matlab і містяться в папці **toolbox\matlab\polyfun**. Всі вбудовані функції орієнтовані на роботу з векторами та матрицями. Тобто вихідна функція повинна задаватися у вигляді вектора-стовпця (матриці) в точках аргументу.

Всі інтерполяційні функції **interp1q**, **interp1**, **interpft**, **spline**, **interpn**, **interp2**, **interp3** СКМ Matlab проводять кускову («ковзаючу») інтерполяцію. Обов'язковими аргументами всіх функцій інтерполяції є вектори-стовпці/матриці **X, Y** вузлів та значень сітчастої функції. Аргумент **Y** може задаватися іменем функції користувача.

Аналогом функції **linterp** з СКМ MathCAD є функція **yi=interp1q(<x>,Y,xi),** яка розраховує кусково-лінійно інтерпольовані значення одномірної функції в точках, які визначені у векторі стовбці **xi**.

Функція

interp1(<x>,Y, xi,<method>,<'extrap'>,<extrapval>,<'pp'>)

дозволяє проводити кускову інтерполяцією декількома способами. Метод інтерполяції визначається опційним аргументом **method** у вигляді забраного в лапки текстового рядка. В функції реалізовані методи **'nearest'**(найближчий), **'linear'**(лінійний) по двох найближчих точках, **'spline'(**кубічний сплайн), **'pchip','cubic'(**кусковий кубічний поліном Ерміта) по чотирьох точках. Сама функція проводить найближчу та лінійну інтерполяції. Для сплайн інтерполяції викликається вбудована функція **spline**, для кубічної інтерполяції – функція **pchip**. Для сплайн та кубічної інтерполяції визначенням аргументу **'extrap'**можливо провести екстраполяцію за межами інтервалу **x**.

Одномірна кубічна сплайн інтерполяція проводиться функцією **y=spline(x,y,xi)**.

Наближення періодичних функцій проводить функція **y=interpft(x,n),** яка виконує одновимірну інтерполяцію з використанням перетворення Фур'є. Функція розраховує вектор **y** довжиною **n,** по значенням вектора періодичної функції **x** довжиною **m (m<=n).**

Проведемо комп'ютерний експеримент з аналізу швидкодії та точності одномірної інтерполяції вбудованими лінійним та кубічним методами, лінійним та квадратичним методами із розв’язанням системи лінійних рівнянь та безпосереднім розрахунком.

Велика швидкодія сучасних комп’ютерів унеможливлює фіксацію часу інтерполяції в одній точці засобами СКМ Matlab. Для компенсації швидкодії застосуємо циклічне повторення дій визначену кількість разів.

Для зручності поля для визначення вихідних даних 4-х вузлів сітчастої функції, кількості повторень розрахунків, виразу сітчастої функції та результати часу, значення інтерполяції, абсолютної похибки моделювання розташуємо в діалоговому вікні (рис. 6.8).

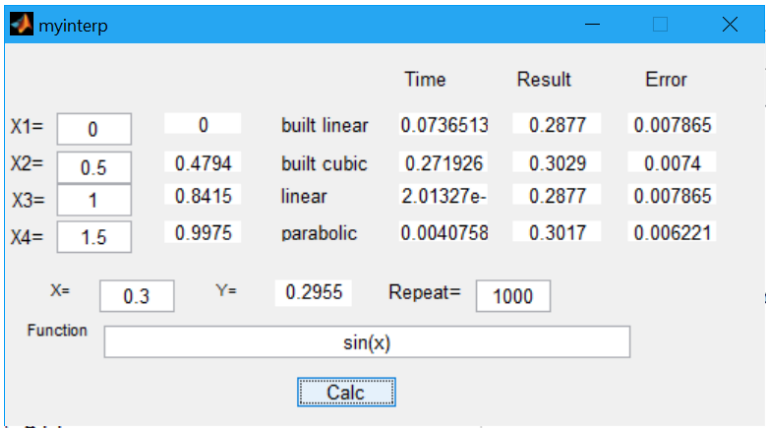


Рисунок 6.8 – Вигляд вікна моделі

РОЗВ’ЯЗАННЯ

Для того, щоб модель ставала працездатною безпосередньо після запуску, конкретизуємо початкові дані. В якості функції оберемо **`sin(x)`**, в якості вузлів сітки – значення **`[0 0.5 1 1.5]`**, в якості точки інтерполяції – **`0.3`**, кількість повторів – **`1000`**.

Для зменшення кількості глобальних змінних в моделі та зменшення витрат на конструювання діалогового вікна експерименту зручно скористатися середовищем **`GUIDE`**. Дані в моделі при цьому будуть передаватися через вбудовану структуру **`handles`**.

В діалоговому вікні розташовані поля введення вузлів сітки **`x1edit`**, **`x2edit`**, **`x3edit`**, **`x4edit`**, поле введення точки інтерполяції **`xedit`**, поле введення виразу функції інтерполяції **`Fedit`**, поле введення кількості

повторень **nedit**, поля виведення результатів значень функції в чотирьох опорних точках , точці інтерполяції, значення вбудованої лінійної **blinrestext**, кубічної інтерполяції **qrestext**, лінійної **linrestext** та параболічної **parrestext** інтерполяції, часу розрахунку **blintimetext, qtimetext, lintimetext, partimetext** та абсолютної похибки кожного виду інтерполяції **blinerrortext, bqerrortext, linerrortext, parerrortext**.

Передача цих та інших даних між підпрограмами проводиться структурою **handles** GUI через поля **x** – для вектору точок аргументу, **y** – вектору значень функції, **point** – для точки інтерполяції, **q** – виразу сітчастої функції, **n** – для кількості повторів. Записуються поля в структуру функцією **guidata**. Ініціюються вихідні значення полів методом **OpeningFcn**.

Кускова інтерполяція проводиться функцією **interp1** з ключем '**linear**' для лінійної інтерполяції, з ключем '**cubic**' – для кубічної.

Лінійна кускова інтерполяції проводиться на наступним виразом:

$$f(p) = y_1 + (y_2 - y_1) \cdot \frac{p - x_1}{x_2 - x_1}.$$

Значення параболічної інтерполяції розраховується в два етапи. На першому по вихідних даних визначаються коефіцієнти A_1, A_2, A_3 параболічного поліному [23]:

$$A_1 = \frac{y_1 \cdot (x_3 - x_2) + y_2 \cdot (x_1 - x_3) + y_3 \cdot (x_2 - x_1)}{d},$$

$$A_2 = \frac{y_1 \cdot (x_2^2 - x_3^2) + y_2 \cdot (x_3^2 - x_1^2) + y_3 \cdot (x_1^2 - x_2^2)}{d},$$

$$A_1 = \frac{y_1 \cdot (x_3 - x_2) \cdot x_3 \cdot x_2 + y_2 \cdot (x_1 - x_3) \cdot x_1 \cdot x_3 + y_3 \cdot (x_2 - x_1) \cdot x_2 \cdot x_1}{d},$$

де **d=(x1-x2)·(x2-x3)·(x3-x1).**

На другому етапі розраховується сам поліном за наступним виразом:

$$f(p) = \sum_{i=1}^{3} A_i \cdot p^{i-1}.$$

Функціонування моделі

Лістинг функції керування наведено в додатку А (рис. А1).

Під час введення значень у відповідні поля введення дані типу «текстовий рядок» переводяться на чисельні дійсні значення та записуються у відповідне поле структури **`handles`**. Обробка функції інтерполяції проводиться функцією **`inline`**, яка виконує дію, прописану в текстовому рядку свого аргументу.

Для зменшення обсягу програмного коду введенням функцій обробки для кожного поля введення створено єдину функцію користувача **`myinit`**. Функція виводить результати значень функції інтерполяції в вузлах сітки та точці інтерполяції у відповідні поля вікна після кожної зміни введених даних.

Час інтерполяції для кожного методу вимірюється вбудованими секундомірами **`tic-toc`**,

Аналіз результатів

Співпадіння похибок вбудованої та «авторської» кускової лінійної інтерполяції показують, що для проведення кускової лінійної інтерполяції в пакеті використана стандартна математична залежність.

Найточніша вбудована кускова кубічна інтерполяція має меншу майже на 20% точність, ніж «авторська» кускова параболічна інтерполяція. Різниця в точності вбудованих кускових інтерполяцій не перевищує 6%.

Для інтерполяції в одній точці вбудовані алгоритми мають суттєво меншу швидкодію в порівнянні з «авторськими». Серед розглянутих алгоритмів лінійна кускова інтерполяція є найшвидшою.

Контрольні запитання

Яку функцію називають сітчастою?

В чому полягає задача наближення функції?

Що таке глобальне наближення?

Що таке локальне наближення?

Що таке нев'язка функції?

В чому полягає умова інтерполяції?

Що таке поліноміальна інтерполяція?

В чому полягає складність функціональної інтерполяції?

В чому полягає «пряма» задача поліноміальної інтерполяції?

В яких випадках рекомендується використання полінома Лагранжа?

Що означає термін «скінчена різниця»?

Що означають терміни інтерполяція «вперед» та «назад»?

Від чого залежить абсолютна гранична похибка глобальної поліноміальної інтерполяції?

В чому полягає «ковзаючи» інтерполяція?

Що таке сплайн?

В чому є переваги сплайн-інтерполяції?

В чому полягає «пряме» розв'язання задачі інтерполяції?

Який тип інтерполяції реалізує функція **`linterp?`**

Який типи інтерполяції реалізує функція **`interp?`**

Де містяться вбудовані функції інтерполяції Matlab?

Який тип інтерполяції проводять вбудовані функції Matlab?

6.6. Апроксимація

Коли вихідні дані x_i і y_i є наближеними і містять похибки, інтерполяційна формула повторюватиме ці помилки буде ідеальним наближенням. На додачу, як правило обсяг вхідних даних досить об'ємний і інтерполяційна формула стає занадто складною.

Якщо аналітично вираз функції **f(x)** невідомий або складний, то виникає важлива практична задача: знайти емпіричну формулу $\varphi(\vec{x}, \vec{a}) \approx f(\vec{x})$ значення якої у вузлах сітки якомога менше відрізнялись від відомих (експериментальних) значень y_i.

В загальній постановці задача є невизначеною. Тому, як правило, вид функції обирають з досить вузького класу функцій, якому повинна належати шукана функція $\varphi(\vec{x}, \vec{a})$ і справа зводиться до відшукання найкращих значень параметрів. Використовують, наприклад, лінійні, степеневі, показникові, логарифмічні функції (табл. 6.3) тощо.

Побудова такої формули складається з двох етапів: обрання типу функції, визначення її параметрів.

Найбільш застосовуваним із методів розв'язання задачі визначення параметрів апроксимаційної функцій є **метод найменших квадратів** (МНК).

МНК призначений безпосередньо для знаходження параметрів $\vec{a} = [a_0, a_2, \dots, a_M]$ апроксимуючої функції $\varphi(\vec{x}, \vec{a})$, вид якої вже обраний.

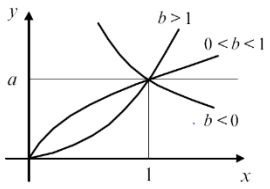
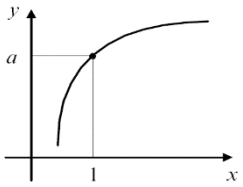
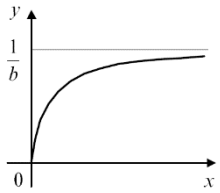
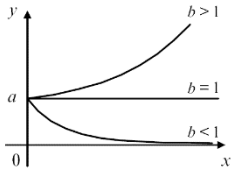
«Цільова» функція $\Phi(\vec{a})$ від параметрів $\vec{a} = [a_0, a_2, \dots, a_M]$ апроксимуючої функції

$$\Phi(\vec{a}) = \sum_{i=0}^{N} (y_i - \varphi(x_i, \vec{a}))^2. \qquad (6.16)$$

показує середньоквадратичне відхилення значень (нев'язок) функції $\varphi(\vec{x}, \vec{a})$ від значень функції f($\vec{x}$) по всій сукупності точок $x_0, x_1, \dots, x_N$.

Параметри функції $\varphi(\vec{x}, \vec{a})$ можна визначити з умови мінімального значення функції $\Phi(\vec{a})$ як значення параметрів, які забезпечують найменшого відхилення $\varphi(\vec{x}, \vec{a})$ від f($\vec{x}$).

Таблиця 6.3. Класи функцій апроксимації

Назва	Вигляд	Залежність
степенева		$y(x) = ax^b + c$
логарифмічна		$y(x) = a\ln(x) + c$ $y(x) = a\ln(x + b) + c$
дрібно-лінійна		$y(x) = \frac{x}{a + bx}$
показова		$y(x) = ae^x + c$
гармонійна		$y(x) = a\sin(bx + c) + d$

Скориставшись необхідними умовами екстремуму функції кількох змінних, одержуємо «нормальну» систему для визначення коефіцієнтів **a_i, i=0 … M.**

$$\frac{d\Phi(\vec{a})}{d\vec{a}} = 2\frac{d\varphi(\vec{a})}{d\vec{a}}\sum_{i=0}^{N}\left(\left(y_i - \varphi(x_i, \vec{a})\right)\right) = 0,$$

$$\begin{matrix} \frac{\partial\Phi(\vec{a})}{\partial a_0} = 0 \\ \frac{\partial\Phi(\vec{a})}{\partial a_1} = 0, \\ \dots \\ \frac{\partial\Phi(\vec{a})}{\partial a_M} = 0 \end{matrix} \qquad \begin{cases} \frac{d\varphi(\vec{a})}{da_0}\sum_{i=0}^{N}\left(\left(y_i - \varphi(x_i, \vec{a})\right)\right) = 0 \\ \frac{d\varphi(\vec{a})}{da_0}\sum_{i=0}^{N}\left(\left(y_i - \varphi(x_i, \vec{a})\right)\right) = 0. \\ \dots \\ \frac{d\varphi(\vec{a})}{da_M}\sum_{i=0}^{N}\left(\left(y_i - \varphi(x_i, \vec{a})\right)\right) = 0 \end{cases} \qquad (6.17)$$

Якщо система (6.17) має єдиний розв'язок, то він дасть значення шуканих коефіцієнтів a_i, $i=0 \dots M$. Складність полягає в тому, що в загальному випадку система є нелінійною.

Наприклад, для сітчастої функції з $N+1$ точок $x_0, x_1, \dots, x_N$, $y_0, y_1, \dots, y_N$, знайдемо значення коефіцієнтів апроксимуючої лінійної функції $\varphi(x, a, b) = ax + b$.

Цільова функція описується як

$$\Phi(x, a, b) = \sum_{i=0}^{N} (y_i - (ax_i + b))^2.$$

Система рівнянь набуває наступного вигляду

$$\frac{d\Phi(a,b)}{d\vec{a}} = 2\frac{d\varphi(x,a,b)}{d\vec{a}} \sum_{i=0}^{N} \big(y_i - (ax_i + b)\big) = 0,$$

$$\begin{cases} \dfrac{\partial \Phi(a,b)}{\partial a} = 0 \\ \dfrac{\partial \Phi(a,b)}{\partial b} = 0 \end{cases}, \qquad \begin{cases} \dfrac{\partial \varphi(x,a,b)}{\partial a} \displaystyle\sum_{i=0}^{N} \big(y_i - (ax_i + b)\big) = 0 \\ \dfrac{\partial \varphi(x,a,b)}{\partial b} \displaystyle\sum_{i=0}^{N} \big(y_i - (ax_i + b)\big) = 0 \end{cases},$$

$$\frac{\partial \varphi(x,a,b)}{\partial a} = \frac{\partial (ax+b)}{\partial a} = x, \quad \frac{\partial \varphi(x,a,b)}{\partial b} = \frac{\partial (ax+b)}{\partial b} = 1,$$

$$\begin{cases} \displaystyle\sum_{i=0}^{N} \Big(x_i \big(y_i - (ax_i + b)\big)\Big) = 0 \\ \displaystyle\sum_{i=0}^{N} \Big(\big(y_i - (ax_i + b)\big)\Big) = 0 \end{cases}, \qquad \begin{cases} a\displaystyle\sum_{i=0}^{N} x_i^2 + b\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} x_i y_i \\ a\displaystyle\sum_{i=0}^{N} x_i + Nb = \sum_{i=0}^{N} y_i \end{cases}.$$

Наведена система є СЛАР другого порядку з двома невідомими, розв'язання якої визначає шукані коефіцієнти апроксимації a, b.

$$\begin{bmatrix} \sum_{i=0}^{N} x_i^2 & \sum_{i=0}^{N} x_i \\ \sum_{i=0}^{N} x_i & N \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} x_i y_i \\ \sum_{i=0}^{N} y_i \end{bmatrix}.$$

Для апроксимуючої квадратичної функції $\varphi(x, a, b, c) = ax^2 + bx + c$ цільова функція описується як

$$\Phi(a, b, c) = \sum_{i=0}^{N} (y_i - (ax_i^2 + bx_i + c))^2.$$

Система рівнянь набуває наступного вигляду

$$\frac{d\Phi(a, b, c)}{d\vec{a}} = 2\frac{d\varphi(x, a, b, c)}{d\vec{a}} \sum_{i=0}^{N} \left(y_i - (ax_i^2 + bx_i + c)\right) = 0,$$

$$\frac{\partial\varphi(x, a, b, c)}{\partial a} = \frac{\partial(ax_i^2 + bx_i + c)}{\partial a} = x^2, \qquad \frac{\partial\varphi(x, a, b, c)}{\partial b} = x,$$

$$\frac{\partial\varphi(x, a, b, c)}{\partial c} = 1,$$

$$\begin{cases} \sum_{i=0}^{N} \left(x_i^2 \left(y_i - \left(ax_i^2 + bx_i + c\right)\right)\right) = 0 \\ \sum_{i=1}^{N} \left(x_i \left(y_i - \left(ax_i^2 + bx_i + c\right)\right)\right) = 0 \\ \sum_{i=0}^{N} \left(y_i - \left(ax_i^2 + bx_i + c\right)\right) = 0 \end{cases} \quad \begin{cases} a\sum_{i=0}^{N} x_i^4 + b\sum_{i=0}^{N} x_i^3 + c\sum_{i=0}^{N} x_i^2 = \sum_{i=0}^{N} x_i^2 y_i \\ a\sum_{i=0}^{N} x_i^3 + b\sum_{i=0}^{N} x_i^2 + c\sum_{i=0}^{N} x_i^1 = \sum_{i=0}^{N} x_i^1 y_i \\ a\sum_{i=0}^{N} x_i^2 + b\sum_{i=0}^{N} x_i + c\sum_{i=0}^{N} x_i^0 = \sum_{i=0}^{N} x_i^0 y_i \end{cases}$$

Наведена система є СЛАР третього порядку з трьома невідомими.

$$\begin{bmatrix} \sum_{i=0}^{N} x_i^4 & \sum_{i=0}^{N} x_i^3 & \sum_{i=0}^{N} x_i^2 \\ \sum_{i=0}^{N} x_i^3 & \sum_{i=0}^{N} x_i^2 & \sum_{i=0}^{N} x_i^1 \\ \sum_{i=0}^{N} x_i^2 & \sum_{i=0}^{N} x_i^1 & \sum_{i=0}^{N} x_i^0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} x_i^2 y_i \\ \sum_{i=0}^{N} x_i^1 y_i \\ \sum_{i=0}^{N} x_i^0 y_i \end{bmatrix}.$$

Розв'язання СЛАР визначає шукані коефіцієнти апроксимації **a, b, c.**

M+1 коефіцієнтів апроксимаційного полінома степеню **M** для табличної функції з **N+1** вузлів знаходяться розв'язанням СЛАР степеню **M+1**. Елементи вектора вільних членів та елементи матриці А СЛАР визначаються як

$$b_i = \sum_{k=0}^{N} x_k^{M-i} y_k, \quad A_{ij} = \sum_{k=0}^{N} x_k^{M+3-i-j}. \qquad (6.18)$$

Випадок поліноміальної апроксимації є важливим, бо більшість «елементарних» функцій зводиться саме до нього:

$$\varphi(x,c,b) = cx^b \rightarrow \ln(\varphi) = \ln(c) + bln(x) = a + bz \; [z = \ln(x) \;\; a = \ln(c)],$$

$$\varphi(x,c,b) = ce^{bx} \rightarrow \ln(\varphi) = \ln(c) + bx = a + bx \; [a = \ln(c)],$$

$$\varphi(x,a,b) = \frac{1}{ax+b} \rightarrow \quad \sigma = \frac{1}{\varphi} = ax + b.$$

В разі застосування функціональної апроксимації $f(\vec{a}, \overrightarrow{x})$ складається система нелінійних рівнянь $\sum_{i=0}^{N} \left(\left(y_i - \varphi(x_i, \vec{a}) \right) \right) \cdot \frac{\partial f(\vec{a})}{\partial \vec{a}} = 0.$

6.7. Засоби апроксимації СКМ MathCAD, Matlab

Засоби MathCAD

СКМ MathCAD має вбудовані засоби для проведення функціональної і поліноміальної апроксимації.

Для поліноміальної апроксимацій максимальний степінь М полінома на відміну від інтерполяції може бути меншим **M≤N-1**.

В [21] для вибору степеню апроксимуючого полінома рекомендовано правило, згідно якому максимальний степінь полінома відповідає порядку кінцевої різниці сітчастої функції, яка є сталою.

Наприклад, для даних, які наведено в табл. 6.4, сталою є друга різниця. Тобто, для глобальної поліноміальної апроксимації досить поліному другого степеню.

Таблиця 6.4 - Аналіз кінцевих різниць

x	1	2	3	4	5	6	7
y	2.2	6.5	12.8	21.1	31.4	43.7	58
Δ_1		4.3	6.3	8.3	10.3	12.3	14.3
Δ_2			2	2	2	2	2

Знаходження коефіцієнтів поліноміальної апроксимації за критерієм мінімізації середньоквадратичного відхилення полінома від дійсних значень **y(x)** зводиться до розв'язання СЛАР.

Побудова полінома апроксимації має наступний алгоритм:

1. Визначення векторів вузлів сітки **x** та сітчастої **y** в **N** вузлах.

2. Визначення матриці Вандермонде **X** коефіцієнтів СЛАР (6.18).

3. Знаходження коефіцієнтів полінома розв'язанням СЛАР (6.18).

Для поліноміальної апроксимації в MathCAD використовується та ж функція, що і для інтерполяції **interp(vs,vx,vy,x,N).** Ознакою проведення саме апроксимації є наявність додаткового аргументу **N**, який визначає степінь апроксимуючого поліному

Для апроксимації вектор допоміжних коефіцієнтів **vs** розраховується функцією **regress(vx, vy, N)**, Результуючий вектор **VS,** починаючи з четвертого елемента містить коефіцієнти $\mathbf{a_i}$ апроксимуючого полінома наближення. Технологія застосування функції **regress** аналогічна проведенню сплайн інтерполяції.

Апроксимація лінійною комбінацією функцій $\mathbf{f_i(x)}$ записується у вигляді багаточлену:

$$\varphi(x) = a_0 f_0(x) + a_1 f_1(x) + \cdots + a_N f_N(x).$$

Функція **linfit(vx,vy,F)** виконує вказане апроксимаційне наближення лінійною комбінацією. Аргументами функції є вектори вузлів та значень сітчастої функції **vx,vy** та вектор-стовпець **F** виразів функцій $\mathbf{f_i(x)}$.

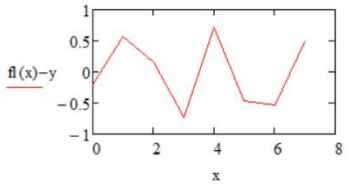
Наприклад, для сітчастої функції з вузлами сітки **[0 1 2 3 4 5 6 7]** та відповідними значеннями сітчастої функції **[4 5 9 20 33 52 73 96]** апроксимація виразом

$$\varphi(x) = ax^2 + be^{-x} + c\frac{1}{1+x} + d$$

може виглядати наступним чином:

$$F(x) := \begin{pmatrix} x^2 & e^{-x} & \frac{x}{1+x} & 1 \end{pmatrix}^T \quad f := \text{linfit}(x, y, F) \quad f = \begin{pmatrix} 1.729 \\ 73.913 \\ 93.51 \\ -70.113 \end{pmatrix}$$

$$fl(x) := f_0 \cdot x^2 + f_1 \cdot \exp(-x) + f_2 \cdot \frac{x}{1+x} + f_3$$



$$\frac{\text{norm}(fl(x) - y)}{\min(y)} = 0.131$$

Розрахунок коефіцієнтів функціональної апроксимації проводиться функцією

y:=genfit (vx,vy,vg,F).

Аргументами функції є вектори-стовпці вузлів та значень сітчастої функції **vx, vy,** вектор початкового наближення для всіх коефіцієнтів **vg,** ім'я функції користувача **F**, яка повертає вектор-стовпець з самої функції апроксимації та її часткових похідних по всіх коефіцієнтах.

Визначення компонентів вектора **F** можна проводити вручну прямим введенням виразів або використання функцій символьного диференціювання (оператор ∇ – **Ctrl+Shift+G**) та **stack**.

Наприклад, для степеневої функції апроксимації $\varphi(x, \vec{a}) = a_0 + a_1 x^{a_2}$ коефіцієнти апроксимації в функції користувача FF можуть бути визначені

– як елементи вектора, шо введені вручну:

$$f(x, a) := a_0 + a_1 \cdot x^{a_2}$$

$$FF(x, a) := \begin{pmatrix} a_0 + a_1 \cdot x^{a_2} \\ 1 \\ x^{a_2} \\ a_1 \cdot x^{a_2} \cdot \ln(x) \end{pmatrix};$$

– як елементи вектора, розраховані функцією **stack**:

$$FF(x,a) := \text{stack}\left(f(x,a), \nabla_a f(x,a)\right) \rightarrow \begin{pmatrix} x^{a_2} \cdot a_1 + a_0 \\ 1 \\ x^{a_2} \\ x^{a_2} \cdot \ln(x) \cdot a_1 \end{pmatrix}$$

– як окремі змінні, що введені вручну:

$$f(x,a,b,c) := a + b \cdot x^c$$

$$FF(x,a,b,c) := \begin{pmatrix} a + b \cdot x^c \\ 1 \\ x^b \\ a \cdot x^b \cdot \ln(x) \end{pmatrix} ;$$

– як окремі змінні, що розраховані функцією **stack**:

$$FF(x,a,b,c) := \text{stack}\left(f(x,a,b,c), \nabla_{a,b,c} f(x,a,b,c)\right) \rightarrow \begin{pmatrix} a + b \cdot x^c \\ 1 \\ x^c \\ b \cdot x^c \cdot \ln(x) \end{pmatrix} .$$

Допускається використання замість імені вектора F імені функції апроксимації **f** у вигляді функції користувача.

Наприклад, $\text{genfit}(x, y, x0, f)$ або $\text{genfit}(x, y, x0, FF)$.

Наприклад, обчислення коефіцієнтів a, b, c функціональної тригонометричної апроксимації $\varphi(x, \overrightarrow{a}) = a_0 + a_1 \sin(a_3 x)$ сітчастої функції з вузлами **(0,1,2,3,4,5)** та значеннями **(1, 1.959, 2.683, 2.995,2.2817, 2.197**) може виглядати наступним чином:

$$fs(x,a,b,c) := a \cdot \sin(b \cdot x) + c \qquad FF(x,a,b,c) := \begin{pmatrix} c + a \cdot \sin(b \cdot x) \\ \sin(b \cdot x) \\ a \cdot x \cdot \cos(b \cdot x) \\ 1 \end{pmatrix}$$

$$\text{genfit}\left[XS, YS, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, fs\right] = \begin{pmatrix} 2 \\ 0.5 \\ 1 \end{pmatrix} \qquad \text{genfit}\left[XS, YS, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, FF\right] = \begin{pmatrix} 2 \\ 0.5 \\ 1 \end{pmatrix} .$$

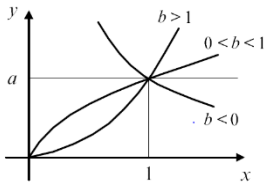
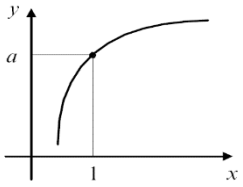
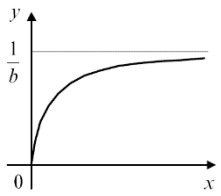
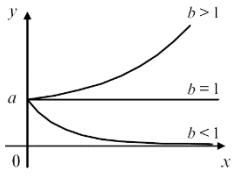
Результат функції **genfit** є вкрай чутливим до початкової точки. Невдале початкове значення може привести до невірної відповіді, або взагалі до неможливості отримати результат.

Наприклад:

$$\text{genfit}\left[\text{XS},\text{YS},\begin{pmatrix}2\\1\\1\end{pmatrix},\text{fs}\right]=\begin{pmatrix}-0.454\\1.401\\2.312\end{pmatrix}\quad \text{genfit}\left[\text{XS},\text{YS},\begin{pmatrix}-1\\-1\\0\end{pmatrix},\text{FF}\right]=$$

Спеціалізовані функції функціональної апроксимації наведено в таблиці 6.5.

Таблиця 6.5. Спеціалізовані функції апроксимації MathCAD

Назва	Вигляд	Залежність
степенева **pwrfit(vx,vy,vg)**		$\varphi(x) = ax^b + c$
логарифмічна **lnfit(vx, vy)** **logfit(vx,vy,vg)**		$\varphi(x) = a \cdot ln(x + b) + c$
дрібно-лінійна		$\varphi(x) = \frac{x}{a+bx}$
показова **expfit(vx,vy,[vg])**		$\varphi(x) = ae^x + c$
гармонійна **sinfit(vx,vy,vg)**		$\varphi(x) = \mathrm{a} \cdot \sin(x + b) + c$

Приклад [12]. Порівняти апроксимацію індикатриси світлодіода АЛ-107Б

α[град]	0	8	15	17	20	30	45	60
F(α)	1.00	0.91	0.70	0.62	0.51	0.33	0.14	0.089

степеневою косінусною функцією $\varphi(a,x)=\cos(x)^a$ та гаусоідою $\varphi(a,x)=\exp(-\frac{x^2}{a^2})$.

РОЗВ'ЯЗАННЯ

Для визначення степеню косинусної апроксимації функцією **genfit** визначимо функцію апроксимації **fc** як функцію користувача з параметром a. Вектор **FC** з функцією апроксимації та її частинними похідними розрахуємо функцією **stack.** Застосуємо функцію **genfit.**

$$fc(a,x):=\left(\cos\left(x\cdot\frac{\pi}{180}\right)\right)^a \qquad FC(a,x):=\text{stack}\left(fc(a,x),\nabla_a fc(a,x)\right)$$

$$a:=\text{genfit}\left(x_{al},y_{al},9.3,fc\right)=14.524$$

$$a:=\text{genfit}\left(x_{al},y_{al},9.3,FC\right)=13.645$$

Результати є насторожуючи ми, бо відповіді для самої функції апроксимації та вектора похідних не збігаються.

Проведемо перевірку безпосереднім розв’язанням системи рівнянь функціональної апроксимації з мінімізацією середньоквадратичного відхилення апроксимуючої функції від вихідної функції.

Розв'язання рівняння блоком **given-find** відповіді не дає.

$$ac:=10$$

Given

$$\left|\overrightarrow{y_{al}-\cos\left(\frac{\pi}{180}\cdot x_{al}\right)^a}\right|=0$$

Find(a) = .

Розпишемо рівняння в явному вигляді:

$$a:=20$$

Given

$$\sum_{i=0}^{imax}\left[\left(y_{al_i}-\cos\left(x_{al_i}\cdot\frac{\pi}{180}\right)^a\right)\cdot\cos\left(x_{al_i}\cdot\frac{\pi}{180}\right)^a\cdot\ln\left(\cos\left(x_{al_i}\cdot\frac{\pi}{180}\right)\right)\right]=0$$

Find(a) = 9.183

Обчислювальний блок видав відповідь 9.183. Проте значення відрізняється від отриманих раніше.

Перевіримо розв'язок функцією **minerr** для рівняння в векторному вигляді:

$$a := 10$$

Given

$$\left| \overrightarrow{y_{al} - \cos\left(\frac{\pi}{180} \cdot x_{al}\right)^a} \right| = 0$$

$$\text{Minerr}(a) = 9.183$$

Функція **minerr** видала відповідь, тотожну з відповіддю функції **find.**

Розрахуємо середньоквадратичні похибки для отриманих значень.

$$\frac{\left| \overrightarrow{y_{al} - fc(14.571, x_{al})} \right|}{imax} = 0.045 \qquad \frac{\left| \overrightarrow{y_{al} - fc(9.186, x_{al})} \right|}{imax} = 0.023$$

$$\frac{\left| \overrightarrow{y_{al} - fc(13.728, x_{al})} \right|}{imax} = 0.041$$

З аналізу похибок видно, що значення, які отримані безпосереднім застосуванням функції **genfit** мають суттєво більші похибки.

Для другого виразу – гаусової апроксимації функція **genfit** взагалі не дає відповіді:

$$fe(ac, x) := e^{-\left(\frac{x}{ac}\right)^2}$$

$$genfit(x_{al}, y_{al}, 100, fe) = \blacksquare$$

Безпосереднє розв’язання рівняння в векторному вигляді функцією **minerr** дозволяє отримати відповідь, яка забезпечує відносну похибку наближення 0.021.

$$ac := 10$$

Given

$$\left| \overrightarrow{y_{al} - \left[e^{-\left(\frac{x_{al}}{ac}\right)^2} \right]} \right| = 0 \qquad \frac{\left| \overrightarrow{y_{al} - fe(26.456, x_{al})} \right|}{imax} = 0.021$$

$$\text{Minerr}(ac) = 26.456$$

З наведеного прикладу очевидним є висновок про ОБОВ'ЯЗКОВУ перевірку результатів функціональної апроксимації функцією **genfit**.

Засоби Matlab

СКМ Matlab має вбудовані засоби тільки для глобальної поліноміальної апроксимації.

Для знаходження коефіцієнтів апроксимації степеневим поліномом призначено функцію **[P<,S,M>]=polyfit(X,Y,N).** Функція проводить глобальний регресійний аналіз по всіх **N** точках вихідних даних. В разі, коли степінь поліному **N** на одиницю менша від розміру векторів аргументів, результатом є квазіінтерполяційний поліном. В інших випадках – апроксимаційний.

Аргументами функції є вектори вузлів сітки **X**, значень функції та степінь поліному **N.**

Результатом є функції вектор **P**, в якому шукані коефіцієнти розміщені в порядку зниження ступеню.

Опціональний елемент відповіді **S** є структурою, яка в своїх полях містить інформацію про результати апроксимації. А саме: коефіцієнт розкладання матриці Вандермонде **R**, ступінь свободи **df**, норму нев'язки **normr**.

Статистичні параметри вектора **X** у вигляді середнього значення та СКВ. можна отримати з опційного елементу відповіді – двоелементного вектору **M.**

Розрахунок степеневої апроксимації в точці (точках) **X** проводиться функцією **[Y<,DELTA>]=polyval(P,X<,S>).** Аргумент вектор **P** містить коефіцієнти поліному. В разі наявності опційного аргументу **S**, можна отримати в векторі результату опційний елемент **DELTA** – СКВ результату.

В стандартній бібліотеці пакета відсутні засоби функціональної апроксимації. Для проведення функціональної апроксимації виробник рекомендує використовувати лінеаризовані моделі з функцією **polyfit** або засоби з додатку оптимізації **Optimization Toolbox** на зразок функцій **lsqcurvefit**, **lsqnonlin**, **fminsearch**.

Лінеаризація нелінійних функцій з двома параметрами проводиться до полінома першого степеня $\varphi(x) = k \cdot x + b$.

Лінеаризація степеневої функції $\varphi(x) = b \cdot x^k$ проводиться логарифмуванням $\ln(\varphi) = k \cdot \ln(x) + \ln(b)$.

Лінеаризація показової функції $\varphi(x) = b \cdot e^{kx}$ теж проводиться логарифмуванням $\ln(\varphi) = k \cdot x + \ln(b)$.

Лінеаризація зворотної дробової функції $\varphi(x) = \frac{1}{kx+b}$ приводиться оберненням $\frac{1}{\varphi(x)} = k \cdot x + b$.

Логарифмічна функція є лінійною $\varphi(x) = k \cdot \ln(x) + b$.

Виклик функції проводиться:

- для степеневої функції – **`polyfit(log(x),log(y),1)`**,
- для показової функції – **`polyfit(x,log(y),1)`**,
- для зворотної лінійної функції – **`polyfit(x,1./y,1)`**,
- для логарифмічної функції – **`polyfit(log(x),y,1)`**.

Відповіддю функції **`polyfit`** з таким викликом буде двоелементний вектор. Перший елемент вектора міститиме значення коефіцієнта **`к`**, другий – значення коефіцієнта **`b`** для логарифмічної та зворотно лінійної, значення **`ln(b)`** – для ступеневої та показової функцій.

Додаток Matlab інтерактивного регресійного аналізу **`Basic Fitting`** дозволяє за даними вихідних векторів вузлів сітки та значень сітчастої функції з робочого простору **`workspace`** визначити коефіцієнти, похибки, відобразити графічно інтерполяцію кубічними сплайнами (**`spline interpolat`**), апроксимацію степеневими поліномами зі степенем від 1 до 10 (**`linear, quadratic, cubic, nth degree polynomial`**), інтерполяцію поліномами Ерміта (**`shape-preserving`**) тощо.

Додаток запускається в вікні графіки **`figure`** з пункту **`Tools`** меню (рис. 6.9).

Наближення проводиться для наборів даних, графіки яких виведено в відповідне вікно **`figure`**. Доступні для обробляння набори відображаються в полі **`Select data`**.

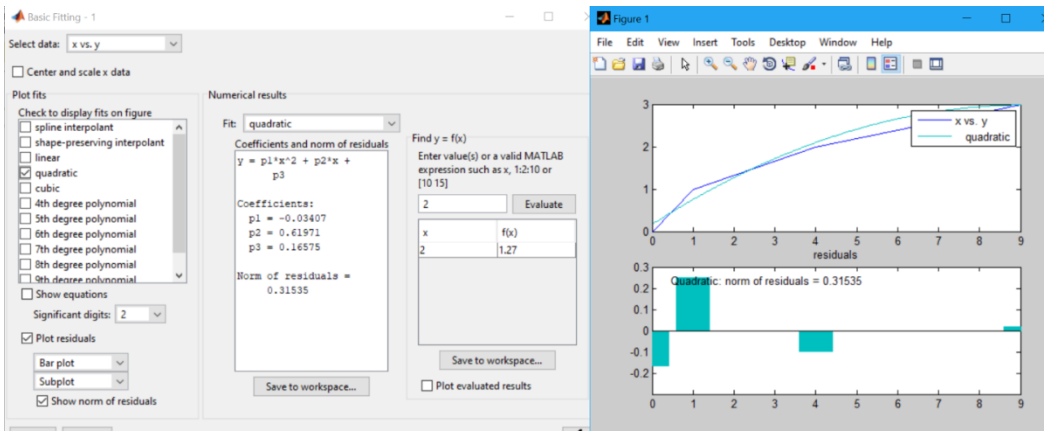


Рисунок 6.9 – Додаток **Basic Fitting**

Тип наближення обирається зі списку **Check to display** на панелі **Plots fits.** Для обраного типу наближення в графічному вікні відображається відповідні графіки з поясненнями та формулами функцій наближення, графіки нев'язок (**residuals**).

На панелі **Numerical results** для обраного зі списку **Fit** виду наближення в полі **Coefficients and norm of residuals:** відображається вираз наближення, значення коефіцієнтів наближення та нев'язок.

Введення значення **x** в поле панелі **Find** та натискання кнопки **Evaluate** виводить в таблицю чисельних значень наближаючої функції для введеного аргументу або діапазону значень аргументу.

Поле вибору **Center and scale x data** керує центруванням даних до середнього значення та нормування до одиничного стандартного відхилення.

Поле вибору **Show equations** керує виведенням формул виразів наближення на графіки.

Поле вибору **Plot residuals** керує виведенням графіку нев'язок.

Поле вибору **Show norm of residuals** керує виведенням чисельного значення норми нев'язки на графік нев'язок.

Приклад [12]. Порівняти апроксимацію індикатриси світлодіода АЛ-107Б

α[град]	0	8	15	17	20	30	45	60
F(α)	1.00	0.91	0.70	0.62	0.51	0.33	0.14	0.089

степеневою косінусною функцією $f(a,x) = \cos(x)^a$ та гаусоідою $f(a,x) = \exp(-\frac{x^2}{a^2})$.

РОЗВ'ЯЗАННЯ

Знаходження значень коефіцієнтів апроксимаційних функцій в Matlab можливо тільки шляхом безпосереднього розв’язання рівняння апроксимації. Для розв'язання нелінійного рівняння апроксимації можливо застосування функції розв’язання рівнянь **fzero** та функції розв’язання систем рівнянь **fsolve**.

В обох функціях аргументом слугує посилання на m-функцію рівняння, яка повинна бути записана у вигляді залежності тільки від коефіцієнтів апроксимації. Рівняння ж функціональної апроксимації залежить також від значень табличної функції. Для виконання обох умов визначимо вектори вузлів сітки та значень сітчастої функції глобальними змінними.

```
function [ yr ] = funur(a )
global xr y
yr=norm(y-cos(xr).^a,2); end
```

Нижче наведено скрипт визначення значення коефіцієнта косинусної апроксимації:

```
clear
global xr y
x=[0 8 15 17 20 30 45 60];
y=[1 0.91 0.70 0.62 0.51 0.33 0.14 0.089];
xr=pi/180*x;
fzero(@funur,1)
fsolve(@funur,1)
a=fsolve(@funur,1);
er=norm(y-cos(xr).^a,2)/(length(x)-1);
fprintf('a= %g erroor=%g',a,er)
```

```
    Exiting fzero: aborting search for an interval
containing a sign change because NaN or Inf function value
encountered during search.
    (Function value at -1309.72 is Inf.)
    Check function or try again with a different starting
value.
    ans =    NaN

    No solution found.
    fsolve stopped because the problem appears regular as
measured by the gradient,but the vector of function values
is not near zero as measured by the default value of the
function tolerance.
    <stopping criteria details>

    a= 9.18274 error=0.0234597
```

З результатів видно, що функція **fzero** не знайшла значення. Причина полягає в тому, що умовою використання функції **fzero** є гарантований перетин осі абсцис функцією рівняння, а апроксимаційна функція має мінімум, який наближається до осі, але не перетинає її.

Функція **fsolve** видала попередження про відсутність кореня рівняння, але визначила точку, в якій апроксимаційна функція максимально наближається до осі Y.

Знайдене функцією **fsolve** значення співпадає з результатом прямого розв'язання рівняння апроксимації в MathCAD.

Для знаходження коефіцієнта гаусової апроксимації замінимо функцію рівняння:

```
function [ yr ] = funur(a )
global xr y
yr=norm(y-exp(-(xr/a).^2),2);
end
```

Нижче наведено скрипт визначення значення коефіцієнта гаусової апроксимації:

```
clc
clear
global xr y
xr=[0 8 15 17 20 30 45 60];
y=[1 0.91 0.70 0.62 0.51 0.33 0.14 0.089];
a=fsolve(@funur,10);
er=norm(y-exp(-(xr/a).^2),2)/(length(xr)-1);
fprintf('a= %g error=%g',a,er)
```

Відповідь функції **fsolve** співпадає з результатом безпосереднього розв'язання рівняння в MathCAD.

a= 26.4555 error=0.0213389.

Контрольні запитання

З яких етапів складається процес побудови апроксимаційної формули?

Який критерій найчастіше застосовується для апроксимації?

Що таке «цільова» функція апроксимації?

В чому полягає важливість поліноміальної апроксимації?

Якою функцією проводиться поліноміальна апроксимація в MathCAD?

Для чого призначено функцію **regress** в MathCAD?

Якою функцією проводиться функціональна апроксимація в MathCAD?

Як оцінити в Matlab СКВ поліноміальної апроксимації?

Для чого слугує застосунок **Basic Fitting?**

7. ЧИСЕЛЬНЕ ІНТЕГРУВАННЯ

Розв'язування багатьох технічних та наукових задач потребує обчислення визначеного інтегралу

$$I = \int_a^b f(x)dx,$$

де **f(x)** – інтегрована на відрізку [**a,b**] функція.

Результатом інтегрування є площа **S** криволінійної трапеції, яка обмежена з боків значеннями відрізку (області) інтегрування [**a,b**], згори – значеннями інтегрованої функції **f(x),** знизу – віссю абсцис OX. Для знаходження такого інтегралу зазвичай використовується формула Ньютона-Лейбніца

$$I = S = \int_a^b f(x)dx = F(a) - F(b), \qquad (7.1)$$

де **F(x)** – первісна функції **f(x).**

Проте на практиці доволі часто зустрічаються випадки, коли первісну неможливо чи важко виразити через елементарні функції або аналітичний вираз первісної має складний і незручний для обчислень вигляд. Крім того, підінтегральна функція може бути задана таблично. В таких випадках використовують чисельні методи інтегрування.

Формули, які використовують для наближеного обчислення інтегралів, називають **квадратурними формулами**. Ідея побудови таких формул полягає в тому, що підінтегральну функцію замінюють іншою, простішою функцією, та знаходять інтеграл від нової функції.

На практиці проводять апроксимацію/інтерполяцію підінтегральної функції **f(x)** іншою функцією **φ(x),** щоб її можна було простіше інтегрувати із наперед визначеною точністю. **Квадратурні формули** базуються на обчисленні функції **f(x)** на сітці $\mathbf{x_i}$**, i=0,1,…,N** проміжку [**a,b**].

Найчастіше застосовуються квадратурні формули такого вигляду:

$$S = \int_a^b f(x)dx = \sum_{k=0}^{N} C_i f(x_i) + \Delta = S_d + \Delta\,, \qquad (7.2)$$

де **Ci** та **xi** називаються коефіцієнтами та вузлами квадратурної формули відповідно. Якщо межі **a, b** проміжку інтегрування належать сітці, то квадратурні формули називають **закритими**, якщо ні – **відкритими**.

Якщо зафіксувати вузли **xi**, залишивши (**N+1**) вільних параметрів **Ci**, то можна вибирати найпростішу сітку вузлів – **рівномірну**. Якщо зафіксувати параметри **Ci**, залишивши вільними параметри **xi**. можна отримати квадратури на **нерівномірній** сітці.

7.1. Квадратурні формули Ньютона-Котеса

Квадратури Ньютона-Котеса є квадратурами з **рівномірною** сіткою.

Якщо в якості наближення функції **f(x)** застосувати інтерполяційний поліном Лагранжа **LN(x)** на рівномірній сітці

$$f(x) = L_N(x) + \Delta_N(x) = \sum_{i=0}^{N} \frac{f(x_i)\omega_{N+1}(x)}{(x-x_i)\omega'_{N+1}(x_i)} + \frac{f^{(N+1)}(\theta)}{(N+1)!}\omega_{N+1}(x),$$

то значення визначеного інтеграла (4.2) визначається як

$$S = \int_a^b f(x)dx = \int_a^b (L_N(x) + \Delta_N(x))dx = \sum_{k=0}^{N} C_i f(x_i) + \Delta = S_d + \Delta,$$

де

$$\Delta = \int_a^b \Delta_N(x)dx, \qquad C_i = \int_a^b \frac{\omega_{N+1}(x)}{(x-x_i)\omega'_{N+1}(x_i)}dx.$$

За умови запису полінома Лагранжа у відносних координатах **x=x0+th**

$$L_N(x = x_0 + th) = L_N(t) = \sum_{i=0}^{N} \frac{f(x_i)\omega_{N+1}(x)}{(x-x_i)\omega'_{N+1}(x_i)} = \sum_{i=0}^{N} \frac{(-1)^{N-i} f_i \prod_{k=0}^{N}(t-k)}{i!\,(N-i)!\,(t-i)}.$$

значення квадратурний коефіцієнт **Ci** можна записати як

$$C_i = \frac{(-1)^{N-i}h}{i!\,(N-1)!}\int_0^N \frac{\prod_{k=0}^{N}(t-k)}{t-i}dt.$$

Замінюючи **`Ci=(b-a)Hi`**, отримуємо формулу **квадратур Котеса**

$$S_d = (b-a)\sum_{i=0}^{N} H_i f(x_i), \tag{7.3}$$

де $H_i = \frac{(-1)^{N-i}}{i!(N-1)!N}\int_0^N \frac{\prod_{k=0}^{N}(t-k)}{t-i}\,dt$ – **коефіцієнти Котеса**. (7.4)

Властивості **коефіцієнтів Котеса**:

- симетричність $H_i=H_{N-i}$;
- нормованість $\sum_{i=0}^{N} H_i = 1$

Для чисельного знаходження значення визначеного інтегралу область інтегрування [**`a,b`**] розбивають на **`N`** підінтервалів, тобто площу криволінійної трапеції **`S`** розглядають як суму криволінійних трапецій **`Si`**, кожна з яких визначена на і-му підінтервалі [$\mathbf{x_{i-1}, x_i}$]

$$S = \sum_{i=1}^{N} s_i.$$

На кожному підінтервалі функцію замінюють на простішу наближену функцію. Такі формули називаються **складеними квадратурними формулами**.

Наприклад, формула Котеса за умови інтерполяції функції прямою (кривою першого порядку) **`N=1`** має наступний вигляд

$$I = \int_{x_i}^{x_{i+1}} f(x)dx \approx h\sum_{i=0}^{1} f_i H_i = h(f_0 H_0 + f_1 H_1).$$

Коефіцієнти Котеса **`H0, H1`** розраховуються (7.4) як

$$H_i = \frac{(-1)^{N-i}}{i!\,(N-1)!\,N}\int_0^N \frac{\prod_{k=0}^{N}(t-k)}{t-i}\,dt.$$

i=0

$$H_0 = \frac{(-1)^{1-0}}{0!\,(1-1)!\,1}\int_0^1 \frac{(t-0)(t-1)dt}{(t-0)}$$

$$= -\int_0^1 \frac{t(t-1)dt}{t}$$

$$= \int_1^0 tdt - \int_1^0 dt = \frac{t^2}{2}|_1^0 - t|_1^0 = 0 - \frac{1}{2} - 0 + 1 = \frac{1}{2},$$

i=1

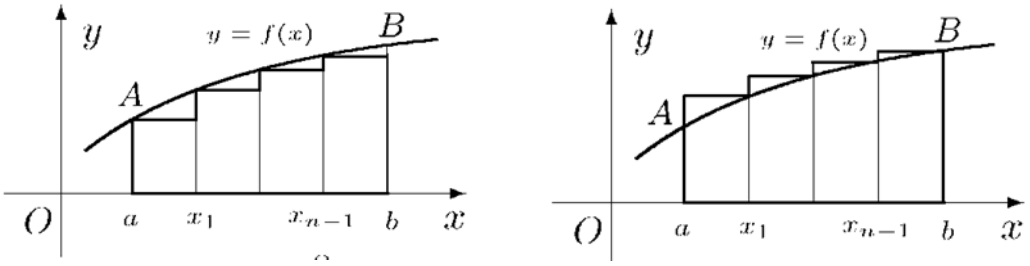
$$H_1 = \frac{(-1)^{1-1}}{1!(1-1)!1}\int_0^1 \frac{(t-0)(t-1)dt}{(t-1)} = \int_0^1 \frac{t(t-1)dt}{t-1} = \int_0^1 tdt = \frac{t^2}{2}|_0^1 = \frac{1}{2}.$$

Квадратурна формула Котеса на підінтервалі [$\mathbf{x_i, x_{i+1}}$] набуває наступного вигляду

$$S = \int_{x_i}^{x_{i+1}} f(x)dx \approx S_d = h(\frac{f_i}{2} + \frac{f_{i+1}}{2}).$$

Формула прямокутників

Розіб'ємо відрізок [**a,b**] на **N** рівних частин з кроком квадратурної формули **h**. На кожному підінтервалі замінимо підінтегральну функцію поліномом першого порядку – горизонтальною прямою, яка проходить через ліву точку границі підінтервалу (рис. 7.1 а) або праву точку границі підінтервалу (рис. 7.1 б).



а б

Рисунок 7.1 – Схема формули прямокутників: а – лівих; б – правих [20]

Геометричний зміст формули прямокутників полягає в тому, що криволінійна трапеція **aABb** замінюється східчастою фігурою, складеною з

прямокутників, одна сторона яких дорівнює кроку квадратури, друга – значенню підінтегральної функції у відповідному вузлі сітки.

Для лівих прямокутників

$$S_i = f(x_i)h = f_i h\,, \quad S_d = \sum_{i=0}^{N-1} f_i h = h\sum_{i=0}^{N-1} f_i. \tag{7.5'}$$

Для правих прямокутників

$$S_i = f(x_{i+1})h = f_{i+1}h, \quad S_d = \sum_{i=0}^{N-1} f_{i+1} h = h\sum_{i=0}^{N-1} f_{i+1}. \tag{7.5''}$$

Формула трапецій

На кожному підінтервалі замінимо підінтегральну функцію поліномом першого порядку – прямою, яка проходить через точки границь підінтервалу (рис. 7.2) тобто проведемо «ковзаючу» лінійну інтерполяцію на підінтервалі $[x_{i-1}, x_i]$.

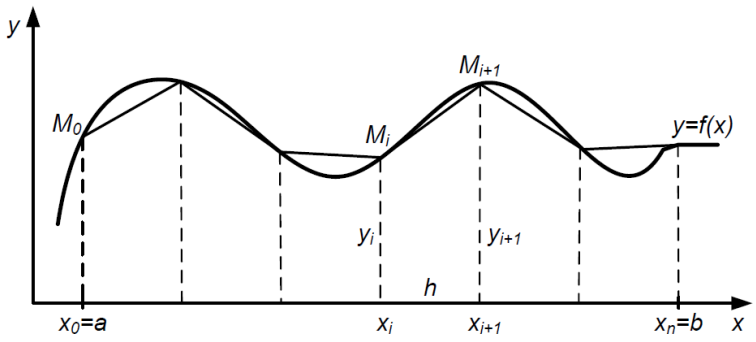


Рисунок 7.2 – Схема формули трапецій [20]

Геометричний зміст формули трапецій полягає в тому, що криволінійна трапеція **aABb** замінюється сумою звичайних трапецій, висота яких дорівнює кроку квадратури, основи – значенню підінтегральної функції на межах підінтервалу.

$$S_i = \frac{f(x_i) + f(x_{i+1})}{2} h, \quad S_d = \sum_{i=0}^{N-1} \frac{f_i + f_{i+1}}{2} h.$$

Вираз для **Sd** є теоретично обґрунтованим, але безпосереднє його використання не економно, бо вимагає виконання зайвих арифметичних дій.

Тотожні перетворення дозволяють отримати більш ефективну форму квадратичної формули трапецій

$$S_d = \sum_{i=0}^{N-1} \frac{f_i + f_{i+1}}{2} h = h\left(\frac{f_0 + f_N}{2} + \sum_{i=1}^{N-1} f_i\right). \tag{7.6}$$

Формула Сімпсона (метод парабол).

Інтервал інтегрування [**a,b**] розбивається на підінтервали [$\mathbf{x_i, x_{i+2}}$] шириною **2h**. Таким чином в кожному підінтервалі знаходиться три точки сітки. На кожному підінтервалі замінимо підінтегральну функцію поліномом другого порядку – параболою, яка проходить через три точки сітки підінтервалу (рис. 7.3) тобто проведемо «ковзаючу» квадратичну інтерполяцію.

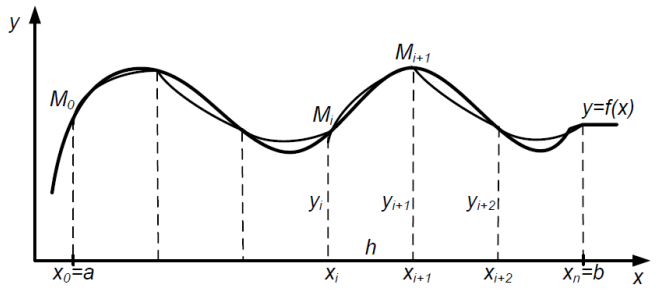


Рисунок 7.3. – Схема формули парабол [20]

Підінтегральну функцію **f(x)** замінимо інтерполяційним многочленом Ньютона другого степеню

$$N_2(x) = f_0 + \frac{\Delta_0^1}{h}(x - x_{0)} + \frac{\Delta_0^2}{2h^2}(x - x_0)(x - x_1).$$

Площа елементарної криволінійної трапеції $\mathbf{S_0}$ на інтервалі [$\mathbf{x_0, x_2}$] обчислюється інтегруванням інтерполяційного поліному Ньютона на визначеному інтервалі.

$$S_0 = \int_{x_0}^{x_2} f(x)dx \approx \int_{x_0}^{x_2} N_2(x)dx =$$

$$= f_0 x|_{x_0}^{x_2} + \frac{\Delta_0}{h}\left(\frac{x^2}{2}|_{x_0}^{x_2} - x_0 x|_{x_0}^{x_2}\right) + \frac{\Delta_0^2}{2h^2}\left(\frac{x^3}{3}|_{x_0}^{x_2} - (x_1 + x_0)\frac{x^2}{2}|_{x_0}^{x_2} + x_0 x_1 x|_{x_0}^{x_2}\right)$$

$$= f_0 2h + \frac{\Delta_0}{h}\frac{4h^2}{2} + \frac{\Delta_0^2}{2h^2}\left(\frac{8h^3}{3} - h\frac{4h^2}{2}\right) = 2hf_0 + 2h\Delta_0 + \frac{h\Delta_0^2}{3} =$$

$$= \frac{h}{3}(6f_0 + 6(f_1 - f_0) + [(f_2 - f_1) - (f_1 - f_0)])$$

$$= \frac{h}{3}(f_0 + 4f_1 + f_2).$$

Відповідно значення інтегралу **Sd** на інтервалі є сумою площин елементарних криволінійних трапецій **Si**

$$S_d = \sum_{i=0}^{N-2} S_i = \frac{h}{3}\sum_{i=0}^{N-2}(f_i + f_{i+1} + f_{i+2}) = \frac{h}{3}(f_0 + 4f_1 + f_2 +$$

$$f_2 + 4f_3 + f_4 + \cdots + f_{N-2} + 4f_{N-1} + f_N) =$$

$$\frac{h}{3}(f_0 + f_N + 2\sum_{i=0}^{N/2-1}(2f_{2i+1} + f_{2i}). \qquad (7.7)$$

Формула модифікованих прямокутників

На кожному підінтервалі замінимо підінтегральну функцію поліномом першого порядку – горизонтальною прямою, яка проходить через центральну точку підінтервалу [x_i, x_{i+1}] (рис. 7.4).

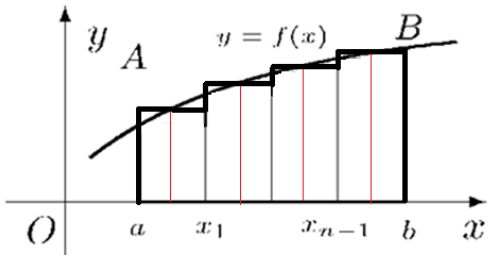


Рисунок 7.4. – Схема формули модифікованих прямокутників

Геометричний зміст формули прямокутників полягає в тому, що криволінійна трапеція **aABb** замінюється східчастою фігурою, складеною з прямокутників, одна сторона яких дорівнює кроку квадратури, друга – значенню підінтегральної функції у точці середини під інтервалу сітки.

$$S_i = f(x_{i+1/2})h = f_{i+1/2}h, \quad S_d = \sum_{i=0}^{N-1} f_{i+1/2}h = h\sum_{i=0}^{N-1} f_{i+1/2}. \quad (7.8)$$

7.2. Точність квадратурних формул Ньютона-Котеса

Формула трапецій

Похибка інтерполяційного поліному (6.11) першого степеня (N=1) для двох сусідніх точок інтервалу [x_{i-1}, x_i] складає

$$R_N(x) \approx \frac{f^{(N+1)}(x*)}{(N+1)!}\omega_{N+1}(x) = f(x) - L_N(x) = \frac{f''(x*)}{2!}(x - x_{i-1})(x - x_i).$$

Абсолютна гранична похибка квадратурної формули Ньютона-Котеса трапецій на підінтевалі [x_{i-1}, x_i] складатиме

$\Delta_h = \int_{x_{i-1}}^{x_i} R_N(x)dx \approx \frac{f''(x*)}{2}\int_{x_{i-1}}^{x_i}(x-x_{i-1})(x-x_i)dx = \frac{f''(x*)}{2}\int_0^h y(y-h)dy =$
$\frac{f''(x*)}{2}\left(\frac{y^3}{3}|_0^h\right) - h\frac{y^2}{2}|_0^h) = \frac{f''(x*)}{2}\left(\frac{2h^3-3hh^2}{6}\right),$

$$\Delta_h \sim f''(x*)\frac{h^3}{12}.$$

Загальна абсолютна гранична похибка квадратурної формули Ньютона-Котеса трапецій на інтервалі [**a,b**] складатиме

$$\Delta_{ab} = \frac{(b-a)}{h}\Delta_h = \frac{b-a}{12}f''(x*)h^2. \qquad (7.11)$$

Формула прямокутників

З урахуванням формули похибки інтерполяційного поліному нульового степеня (N=0) для однієї точки на підінтевалі [$\mathbf{x_{i-1}, x_i}$]

$$R_N(x) \approx \frac{f^{(N+1)}(x*)}{(N+1)!}\omega_{N+1}(x) = f(x) - L_N(x) = \frac{f'(x*)}{1!}(x-x_i).$$

Абсолютна гранична похибка квадратурної формули Ньютона-Котеса прямокутників на підінтевалі [$\mathbf{x_{i-1}, x_i}$] складатиме

$\Delta_h = \int_{x_{i-1}}^{x_i} R_N(x)dx \approx f'(x*)\int_{x_{i-1}}^{x_i}(x-x_i)dx = f'(x*)\int_0^h ydy =$
$f'(x*)\left(\frac{y^2}{2}|_0^h\right) = f'(x*)\left(\frac{h^2}{2}\right),$

$$\Delta_h \sim f'(x*)\frac{h^2}{2}.$$

Аналогічного результату можна отримати урахуванням лінійного члена розкладання в ряд Тейлора функції **f(x)** в околі точки $\mathbf{x_i}$ (формула Лагранжа)

$$f(x) \approx f(x_i) + f'(x*)(x-x_i).$$

$\int_{x_{i-1}}^{x_i} f(x)dx \approx \int_{x_{i-1}}^{x_i} f(x_i)dx + \int_{x_{i-1}}^{x_i} f'(x*)(x-x_i)dx = f_{i-1}h +$
$f'(x*)\frac{(x-x_i)^2|_{x_{i-1}}^{x_i}}{2} = f_{i-1}h + f'(x*)\frac{h^2}{2}, \qquad \Delta_h \approx f'(x*)\frac{h^2}{2}.$

Загальна абсолютна гранична похибка квадратурної формули Ньютона-Котеса прямокутників на інтервалі [**a,b**] складатиме

$$\Delta_{ab} = \frac{(b-a)}{h}\Delta_h = N\Delta_h = \frac{b-a}{2}f'(x*)h. \qquad (7.12)$$

Формула Сімпсона

Особливістю формули Сімпсона є те, що інтерполяційною функцією слугує не квадратична функція-парабола, за назвою формули, а кубічний поліном третього степеня з парними коренями.

З урахуванням формули похибки інтерполяційного поліному третього степеня (N=3) для чотирьох точок, з яких дві точки співпадають

$$R_N(x) \approx \frac{f^{(N+1)}(x*)}{(N+1)!}\omega_{N+1}(x) = f(x) - L_N(x)$$
$$= \frac{f^{IV}(x*)}{4!}(x - x_{i-1})(x - x_i)^2(x - x_{i+1}).$$

Абсолютна гранична похибка квадратурної формули Сімпсона на підінтевалі [**xi-1,xi**] складатиме

$\Delta_h = \int_{x_{i-1}}^{x_{i+1}} R_N(x)dx \approx \frac{f^{IV}(x*)}{24}\int_{x_{i-1}}^{x_{i+1}}(x - x_{i-1})(x - x_i)^2(x - x_{i+1})xdx = \frac{f^{IV}(x*)}{24}\int_{-h}^{h}(y - h)y^2(y + h)dy = \frac{f^{IV}(x*)}{12}\left(\frac{y^5}{5}|_0^h\right) - h^2\frac{y^3}{3}|_0^h) = \frac{f^{IV}(x*)}{12}\left(\frac{3h^5 - 5h^5}{15}\right)$,

$$\Delta_h \sim f^{IV}(x*)\frac{h^5}{90}.$$

Загальна абсолютна гранична похибка квадратурної формули Сімпсона на інтервалі [**a,b**] складатиме

$$\Delta_{ab} = \frac{(b-a)}{h}\Delta_h = \frac{b-a}{90}f^{IV}(x*)h^4. \qquad (7.13)$$

Формула модифікованих прямокутників

Розкладання в ряд Маклорена функції **f(x)** має наступний вигляд

$$f(x) \approx f(0) + tf'(0) + \frac{t^2}{2}f''(0).$$

Для лівої точки інтервалу значення відносної координати **0>t>-1/2** є від'ємним, для правої **0<t<1/2** – додатним.

Значення визначеного інтегралу на інтервалі **[-h/2,h/2]**

$$I_{h/2} = \int_0^{h/2} (f_0 + tf'(0) + \frac{t^2}{2} f''(0))\, dt = \frac{hf_0}{2} + f'(0)\frac{h^2}{8} + f''(0)\frac{h^3}{2 \cdot 3 \cdot 8},$$

$$I_{-h/2} = \int_0^{h/2} (f_0 - tf'(0) + \frac{t^2}{2} f''(0))\, dt = \frac{hf_0}{2} - f'(0)\frac{h^2}{8} + f''(0)\frac{h^3}{48},$$

$$I_h = I_{\frac{h}{2}} + I_{-\frac{h}{2}} = hf_0 + f''(0)\frac{h^3}{24}.$$

Абсолютна гранична похибка квадратурної формули модифікованих прямокутників Ньютона-Котеса на підінтевалі [$\mathbf{x_{i-1}, x_i}$] складатиме

$$\Delta_h \sim f''(0)\frac{h^3}{24}.$$

Абсолютна гранична похибка квадратурної формули модифікованих прямокутників Ньютона-Котеса на інтервалі [$\mathbf{x_{i-1}, x_i}$] складатиме

$$\Delta_{ab} = \frac{(b-a)}{h}\Delta_h = \frac{b-a}{24} f''(x*)h^2. \qquad (7.14)$$

Зведені результати похибок квадратурних формул Ньютона-Котеса наведено в табл. 7.1.

Таблиця 7.1 – Похибки квадратур Ньютона-Котеса

Квадратура	Література	Розрахунок
Трапеції	$\frac{b-a}{12} f''(x^*)h^2$	$\frac{b-a}{12} f''(x^*)h^2$
Прямокутники	$\frac{b-a}{12} f''(x^*)h^2$	$\frac{b-a}{2} f'(x^*)h$
Сімпсона	$\frac{b-a}{90(180{,}1440{,}2440)} f^{IV}(x^*)h^4$	$\frac{b-a}{90} f^{IV}(x^*)h^4$
Модифікованих прямокутників	-	$\frac{b-a}{24} f''(x^*)h^2$

Абсолютна похибка квадратур Ньютона-Котеса прямо пропорційна довжині інтервалу інтегрування (**b-a**), відповідному степеню кроку інтегрування **h** та значенню похідної підінтегральної функції **f(x)**.

Треба відмітити, що отримані значення похибок не співпадають зі значеннями, які можна зустріти в літературі. До значень, які не підтверджуються обґрунтуванням слід відноситися з осторогою.

7.3. Чисельне інтегрування з наперед визначеною точністю

Для розв'язання задач інтегрування можна було б скористатися оцінками п.7.2, але для це вимагає оцінки максимального значення модуля похідної на відрізку , що може виявитися достатньо важким або зовсім неможливим.

Для практичного вжитку можна скористатися **методом подвійного рахування**.

Розглянемо значення інтегрування за квадратурними формулами за умови зменшення кроку інтегрування в два рази:

$$S = S_d(h) + \Delta(h) = S_d\left(\frac{h}{2}\right) + \Delta\left(\frac{h}{2}\right). \qquad (7.15)$$

З урахуванням виразів п. 7.2 відношення **k** абсолютних граничних похибок квадратур Ньютона-Котеса для кроків інтегрування **h** та **h/2** складатиме

$$k = \frac{\Delta(h)}{\Delta(\frac{h}{2})} = \begin{cases} \dfrac{f'(x^*)h}{2}\dfrac{2}{f'(x^*)\dfrac{h}{2}} = 2, \text{формула прямокутників} \\ \dfrac{f''(x^*)h^2}{12}\dfrac{12}{f''(x^*)(\frac{h}{2})^2} = 4, \text{формула трапецій} \\ \dfrac{f^{IV}(x^*)h^4}{90}\dfrac{90}{f^{IV}(x^*)(\frac{h}{2})^4} = 16, \text{формула парабол} \\ \dfrac{f''(x^*)h^2}{24}\dfrac{24}{f''(x^*)(\frac{h}{2})^2} = 4, \text{формула модифік. прямокутників} \end{cases}$$

З виразу (7.15) отримуємо

$$S_d\left(\frac{h}{2}\right) - S_d(h) = \Delta(h) - \Delta\left(\frac{h}{2}\right) = \Delta\left(\frac{h}{2}\right)(k-1) = \Delta(k-1). \qquad (7.16)$$

Тобто, різниця між значеннями чисельного інтегралів, які обчислені з кроками **h** та **h/2**, пропорційна абсолютній погрішності Δ помноженій на коефіцієнт квадратури **k**.

Обчислювальна схема обчислення визначених інтегралів із наперед визначеною абсолютною погрішністю Δ складається з наступних дій:

1. Задати початкове значення **N(h)** та обчислити значення інтеграла за обраною квадратурою **S1.**

2. Збільшити значення **N вдвічі** (зменшити **вдвічі** крок **h**) та обчислити значення інтеграла **S2**.

3. Порівняти модуль різниці |**S2-S1**| із добутком заданої погрішності **Δ** та **(k-1)** .

4. Кроки 2 – 3 повторити доти, доки не виконається умова **|S2-S1|< Δ·(k-1)**

7.4. Квадратура Ричардсона – Ромберга

Квадратура Ричардсона-Ромберга є алгоритмічною модифікацією формули трапецій, яка забезпечує пришвидшення обчислень.

Для метода трапецій розкладання похибки чисельного інтегрування в степеневий ряд показує наявність в останньому тільки парних степенів

$$\Delta_d(h) = a_1 h^2 + a_2 h^4 + a_3 h^6 + \cdots = \sum_{i=1}^{\infty} a_i h^{2i}.$$

Значення коефіцієнтів розкладання **ai** не залежать від значень кроку інтегрування **h** та положення вузлів сітки і можуть бути визначені як

$$a_i = C_i \left(f^{(i-1)}(b) - f^{(i-1)}(a) \right),$$

де **Ci** – коефіцієнт розкладання в ряд Тейлора функції

$$\varphi(x) = \frac{x}{2} \frac{e^x + 1}{e^x - 1}.$$

Розглянемо значення формули трапецій для кроків **h** та **h/2**

$$S_d(h) = S + \Delta_d(h) = S + a_1 h^2 + a_2 h^4 + a_3 h^6 + \cdots$$

$$S_d(h/2) = S + \Delta_d(h/2) = S + a_1(\frac{h}{2})^2 + a_2(\frac{h}{2})^4 + a_3(\frac{h}{2})^6 + \cdots$$

Віднімемо від значення інтеграла для крока **h** значення інтеграла для кроку **h/2**

$$4S_d\left(\frac{h}{2}\right) - S_d(h)$$

$$= 4S + \frac{4a_1h^2}{4} + \frac{4a_2h^4}{16} + \frac{4a_3h^6}{64} + \cdots - S - a_1h^2 - a_2h^4 - a_3h^6$$

$$- \cdots = 3S - \frac{3a_2h^4}{4} - \frac{7a_3h^6}{8} - \cdots$$

Звідки

$$S = \frac{4S_d\left(\frac{h}{2}\right) - S_d(h)}{3} + \Delta', \qquad \Delta' = a_2\frac{h^2}{4} + a_3\frac{7h^2}{24} + \cdots$$

Тобто, значення інтеграла зі зменшеною погрішністю (Δ'<Δ) можна отримати не безпосереднім розрахунком за рахунок зменшення кроку інтегрування, а простими арифметичними перетвореннями значень інтегрування, які зроблені на попередніх кроках.

Ітераційна формула Ромберга має наступний вигляд

$$I_{dk}(h) = \frac{2^{2k}I_{d\,k-1}\left(\frac{h}{2}\right) - I_{d\,k-1}(h)}{2^{2k}-1}. \qquad (7.17)$$

Послідовність дій інтегрування за методом Ричардсона-Ромберга наведено в табл. 7.3.

Таблиця 7.3 – Алгоритм Ричардсона-Родберга

i		k=1	k=2	k=3	k=4
1	Id(h)				
2	Id(h/2)	Id1(h)			
3	Id(h/4)	Id1(h/2)	Id2(h)		
4	Id(h/8)	Id1(h/4)	Id2(h/2)	Id3(h)	
5	Id(h/16)	Id1(h/8)	Id2(h/4)	Id3(h/2)	Id4(h)

7.5. Квадратура Гауса

Квадратура Гауса є квадратурою з нерівномірною сіткою.

Квадратура Гауса була розроблена для інтегрування добутку функцій **ρ(x)·f(x)**

$$I_g = \int_{-1}^{1} \rho(x) f(x) dx \approx \sum_{i=1}^{N} C_i f_i \qquad (7.18)$$

за умови, що для функції **f(x)=x^α** квадратура забезпечить точне значення визначеного інтеграла для всіх показників степеню **α=0…αmax**. При цьому максимальне значення αmax повинно складати **2N-1**, а не визначатися відношенням інтерполяції **αmax=N-1**.

Пояснимо зміст квадратури Гауса на прикладі інтегрування степеневої функції

$$\int_{-1}^{1} x^{\alpha} dx = \sum_{i=1}^{N} C_i f_i.$$

Для однієї точки **N=1** максимальний показник степеню **αmax=2N-1=2-1=1.** Формула має значення для **α=0…αmax=0…1 :**

α=0,

$$\int_{-1}^{1} x^0 dx = \int_{-1}^{1} dx = 2 = \sum_{i=1}^{1} C_i x_i^0 = C_1, \qquad C_1 = 2.$$

α=1,

$$\int_{-1}^{1} x dx = \frac{x^2}{2} \Big|_{-1}^{1} = 0 = \sum_{i=1}^{1} C_i x_i = C_1 x_1, \qquad x_1 = 0.$$

Відповідно значення інтеграла (рис. 7.4)

$$I_g = 2f(0).$$

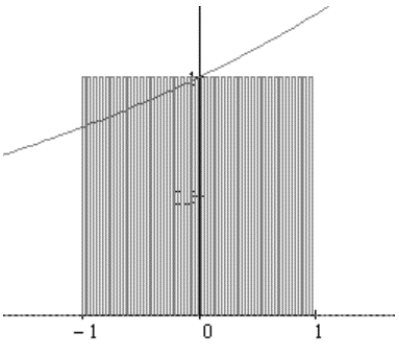


Рисунок 7.4 – Квадратура Гауса для N=1

Для двох точок **N=2** максимальний показник степеню **αmax=2N-1=3.** Формула має значення для **α=0…αmax=0…3.**

$$\alpha=0,\quad \int_{-1}^{1} x^0 dx = \int_{-1}^{1} dx = 2 = \sum_{i=1}^{2} C_i x_i^0, \qquad C_1 + C_2 = 2,$$

$$\alpha=1,\quad \int_{-1}^{1} x dx = \frac{x^2}{2}\Big|_{-1}^{1} = 0 = \sum_{i=1}^{2} C_i x_i, \qquad C_1 x_1 + C_2 x_2 = 0,$$

$$\alpha=2,\quad \int_{-1}^{1} x^2 dx = \frac{x^3}{3}\Big|_{-1}^{1} = \frac{2}{3} = \sum_{i=1}^{2} C_i x_i^2, \qquad C_1 x_1^2 + C_2 x_2^2 = \frac{2}{3},$$

$$\alpha=3,\quad \int_{-1}^{1} x^3 dx = \frac{x^4}{4}\Big|_{-1}^{1} = 0 = \sum_{i=1}^{2} C_i x_i^3. \qquad C_1 x_1^3 + C_2 x_2^3 = 0.$$

Знаходження значень коефіцієнтів **C_1, C_2** та координат вузлів **x_1, x_2** потребує розв'язання СЛАР:

$$C_1 = C_2 = 1,\quad x_1 = -\frac{1}{\sqrt{3}},\quad x_2 = \frac{1}{\sqrt{3}}.$$

Відповідно значення інтеграла (рис. 7.5)

$$I_g = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right).$$

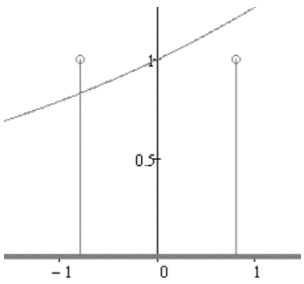


Рисунок 7.5 – Квадратура Гауса для N=2

Для знаходження значень коефіцієнтів **Ci** на вузлів сітки **xi, i=1…N** застосовують наближення функції **f(x)** інтерполяційним поліномом $\mathbf{F_i(x)}$

$$C_i = \int_{-1}^{1} \rho(x) F_i(x) dx,$$

де індекс **i** визначає кількість точок, по яких будується поліном.

Інтерполяційний поліном $\mathbf{F_i(x)}$ має бути ортогональним з вагою **ρ(x)** будь-якому поліному $\mathbf{Q_i(x)}$ зі степенем, меншим за **i** :

$$\int_{-1}^{1} F_i(x) \rho(x) Q_i(x) dx = 0.$$

Таким поліномом є поліном-функція Лежандра, який є розв'язанням диференційного рівняння

$$(1 - x^2)y'' - 2xy' + N(N+1) = 0.$$

Для простого практичного випадку **ρ(x)=1** та задачі квадратури Гауса $\mathbf{Q(x)=x^{\alpha}}$ умова ортогональності набуває наступного вигляду

$$\int_{-1}^{1} L_i(x) x_i^{\alpha} dx = 0 = \sum_{i=1}^{N} C_i \, L_i(x_i) x_i^{\alpha}, \qquad L_i(x_i) = 0.$$

Тобто, вузли квадратури Гауса визначаються як корені поліномів Лежандра.

Способи визначення поліному Лежандра

Поліном Лежандра можна визначити:

- через гіпергеометричну функцію

$$F_i(x) = L_i(x) = \Gamma\left(i+1, -i, 1, \frac{1-x}{2}\right),$$

- за допомогою диференційної формули

$$F_i(x) = L_i(x) = \frac{1}{2^i i!} \frac{d^{(i)}}{dx^i} (x^2 - 1)^i.$$

Наприклад,

$$L_0(x) = \frac{1}{2^0 0!} = 1,$$

$$L_1(x) = \frac{1}{2^1 1!}\frac{d(x^2-1)}{dx} = \frac{1}{2}2x = x, \quad .$$

$$L_2(x) = \frac{1}{2^2 2!}\frac{d^2(x^2-1)^2}{dx^2} = \frac{1}{8}\frac{d}{dx}(2(x^2-1)2x) = \frac{1}{2}\frac{d(x^3-x)}{dx} = \frac{3x^2-1}{2}$$

– за допомогою рекурентної формули

$$(i+1)L_{i+1}(x) = (2i+1)xL_i(x) - iL_{i-1}(x). \qquad (7.19)$$

Наприклад

$$i = 1, \quad (1+1)L_2(x) = (2+1)xL_1(x) - 1L_0(x),$$

$$2L_2(x) = 3xx - 1,$$

$$L_2(x) = \frac{3x^2-1}{2}.$$

Для знаходження значення коефіцієнта **Ci** використовують вираз

$$C_i = \frac{2(1-x_i^2)}{N^2 L_{N-1}^2(x_i)}. \qquad (7.20)$$

Наприклад, для двох вузлів інтегрування (**N=2**)

$$C_1 = \frac{2(1-x_1^2)}{2^2 L_1^2(x_1)}, \;\; L_2(x) = 0, \qquad x = \pm\frac{1}{\sqrt{3}}, \;\; C_1 = \frac{2(1-(\frac{-1}{\sqrt{3}})^2)}{4(\frac{-1}{\sqrt{3}})^2} = \frac{2/3}{2\cdot 1/3} = 1,$$

$$C_2 = \frac{2(1-x_2^2)}{2^2 L_1^2(x_2)}, \qquad C_2 = \frac{2(1-(\frac{1}{\sqrt{3}})^2)}{4(\frac{1}{\sqrt{3}})^2} = \frac{2/3}{2\cdot 1/3} = 1.$$

Результати співпадають із наведеним вище прикладом прямого знаходження коефіцієнтів та вузлів розв'язанням СЛАР.

Квадратура Гауса забезпечує значно меншу похибку в порівнянні з квадратурами Ньютона-Котеса

$$\Delta_{\boldsymbol{g}}(N) = \frac{N(b-a)^{2N+1} f^{(2N)}(x)}{(2N+1)(2N!)^3}. \qquad (7.21)$$

Для практичного вжитку можна рекомендувати наступну обчислювальну схему квадратури Гауса:

1. Приведення інтеграла до нормованого діапазону

$$\int_a^b f(x)dx = \begin{bmatrix} x & z \\ a & -1 \\ b & 1 \end{bmatrix} \quad z = \frac{2x-(b+a)}{b-a} = \frac{b-a}{2}\int_{-1}^{1} f(z)\,dz.$$

2. По заданій кількості точок **N** визначення коренів поліному Лежандра $\mathbf{L_N(x)}$ (4.19).
3. Визначення коефіцієнтів **Ci** (4.20).
4. Розрахунок квадратури Гауса (4.18).

7.6. Засоби інтегрування СКМ MathCAD, Matlab

В MathCAD інтегрування проводиться в один клік введення звичних символів з панелі розширених операторів.

Інтегрування за замовчанням виконується методом Ромберга з точністю 0.001. Змінити метод інтегрування можна в контекстному меню оператора інтегрування (рис.7.6). Змінити точності проводиться перевизначенням змінної TOL.

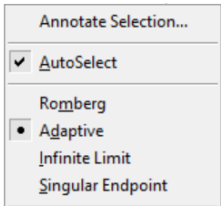


Рисунок 7.6 – Контекстне меню інтегрування

Інтегрування в Matlab потребує програмування з використанням вбудованих функцій. Для чисельного обчислення однократного визначеного інтеграла можуть використовуватися функції **trapz, quad, quadl**.

Функція **trapz(<X>,Y< dim >)** обчислює інтеграл від вектора або матриці **Y** методом трапецій. Опціональний вектор **X** визначає значення аргументу, в яких буде обчислюватися інтеграл. В разі відсутності **X** вважається, що крок інтегрування дорівнює одиниці. Опціональний параметр **dim** визначає рядки або стовпці матриці **Y**, для яких буде проводитися інтегрування.

Наприклад, виклик розрахує інтеграл методом трапецій по чотирьох точках з кроком 1.

```
>> y=[1,2,3,4];
>> trapz(y)
ans = 7.5000
```

Виклик розрахує інтеграл для функції **cos(x)** для аргументу **X** в діапазоні від 0 до π/2 з кроком π/70.

```
>> X=0:pi/70:pi/2; Y=cos(X); Z = trapz(Y)
Z = 22.2780
```

Функція потребує додаткового аналізу точності результатів інтегрування, тому не рекомендується для вжитку.

Функція **quad(fun,a,b,<tol,trace,P1,P2,…>)** обчислює інтеграл від функції **fun** методом парабол Сімпсона. Параметри **a, b** визначають межі інтегрування. Опціональними параметрами є **tol,trace,P1,P2**. Параметр **tol** визначає точність інтегрування, яка за замовчанням встановлена 10^{-6}. Параметр **trace** дозволяє вивести графік про хід інтегрування в разі присвоєння ненульового значення. Параметри **P1,P2** є параметрами користувача, які можна передати до підінтегральної функції.

Підінтегральна функція визначається текстовим рядком або посиланням на функцію користувача. Функція повинна бути записана із використанням векторних поелементних дій.

Функція **quadl** проводить інтегрування методом Гауса-Лобатто. Синтаксис функції є аналогічним синтаксису функції **quad**.

Наприклад, виклик з текстовим рядком:

```
>> quad('x.*exp(-x.^0.8)+0.2',0,8)
ans =3.1604
```

виклик з посиланням на функцію користувача:

```
function y=mfun(x)
y=x.*exp(-x.^0.8)+0.2;
```

```
>> q=quad(@mfun,0,8)
q =3.1604
```

виклик **inline** з функцією:

```
» f=inline(('x.*exp(-x.^0.8)+0.2',xt')
Inline function:
f(x) ='x*exp(-x^0.8)+0.2
» quad(f,0,8)

ans =3.1604
```

Для інтегрування можуть також бути використані функції **quadgk** (обчислення методом Гауса-Кронрода), **quadv** (обчислення методом Сімпсона для векторних функцій скалярного аргументу) та засоби символьних обчислень з додатків **Mupad, Symbolic Toolbox**,.

В новітніх версіях для інтегрування рекомендовано застосовувати функції **integral, polyint.**

Функція **integral(fun,a,b <,Name,Value,...>)** виконує розрахунок однократного визначеного інтегралу від функції **fun** з границями **a,b**.

Підпрограма розрахунку підінтегральної функції **fu** має бути написана у формі, яка дозволяє використання в якості аргументу вектора.

Опціональні параметри **Name,Value** – пара «властивість-значення» визначають додаткові умови інтегрування.

Властивість **'AbsTol'** визначає значення абсолютної похибки інтегрування. Значення є додатним числом.

Властивість **'RelTol'** визначає значення відносної похибки інтегрування. Значення є додатним числом.

Властивість **'ArrayValued'** дозволяє розрахувати значення інтеграла для кількох однотипних функцій. Значенням є **false** (за замовчанням)/ **true**. Наприклад, інтегрування функцій **cos(x), cos(2x),cos(3x),cos(4x),cos(5x)** може виглядати наступним чином:

```
f=@(x)cos((1:5)*x);

q=integral(f,0,1,'ArrayValued',true)

q=  0.8415    0.4546    0.0470   -0.1892   -0.1918
```

Властивість '**Waypoints**' дозволяє явно включити визначені точки в інтегрування. розрахувати значення інтеграла для кількох однотипних функцій. Значення є вектором-рядком. Елементами рядка є визначені точки.

Наприклад, інтегрування функції $f(x) = \frac{1}{2x-3}$ по трикутному контуру від 0 до 1+1i та 1-1i може мати наступний вигляд:

```
f=@(x)1./(2*x-3);
q=integral(f,0,1,'Waypoints',[1+1i,1-1i])
q=-0.5493 + 0.0000i
```

Функція **polyint(p <,k>)** виконує розрахунок коефіцієнтів результату інтегрування поліному.

Поліном визначється у вигляді вектора-рядка p.

Опціональний аргумент **Name,Value** є числом, яку визначає сталу інтегрування.

Результатом є вектор-рядок з коефіцієнтами.

Наприклад, використання функції **polyint** для поліному $y(x) = 3x^4 - 2x^2 + x - 5$ $(\int(3x^4 - 2x^2 + x - 5)\,dx = \frac{3}{5}x^5 - \frac{2}{3}x^3 + \frac{1}{2}x^2 - 5x + k)$ може мати наступний вигляд:

```
>> polyint([3 0 -2 1 -5])
ans=0.6000   0   -0.6667   0.5000   -5.0000 0
```

Контрольні запитання

Що таке квадратурні формули?

До яких квадратур належать формули Ньютона-Котеса?

Яку інтерполяцію використано в квадратурі Котеса?

Що таке складені квадратурні формули?

В чому різниці формули прямокутників та модифікованих прямокутників?

В чому полягає геометричний зміст формули прямокутників?

Що виступає інтерполюючою функцією в формулі трапецій?

В чому полягає геометричний зміст формули трапецій?

В чому полягає геометричний зміст формули Сімпсона?

Для чого використовується метод подвійного рахування?

Для якої квадратурної формули застосовується метод Ромберга?

Яким чином визначаються вузли інтегрування в квадратурі Гауса?

Як змінити метод інтегрування в MathCAD?

Який метод інтегрування реалізує функція **`trapz`**?

Який метод інтегрування реалізує функція **`quad`**?

Який метод інтегрування реалізує функція **`quad1`**?

8. ЧИСЕЛЬНЕ ДИФЕРЕНЦІЮВАННЯ

Похідною функції **y=f(x)** в точці **x** називається границя

$$f'(x) = \frac{df}{dx} = \lim_{dx \to 0} \frac{f(x+dx) - f(x)}{dx}.$$

Чисельне диференціювання застосовується шляхом знаходження замість аналітичного значення похідної **f'(x)** наближених значень в заданих точках коли аналітичний вид функції **f(x)** невідомий, тобто функція визначена неявно або складний, коли функція задана таблицею своїх значень $\mathbf{y_i = f(x_i)}$.

Запис формул диференціювання через функцію **f(x)** відповідає постановці задачі диференціювання, коли її аналітичний вигляд відомий, тобто значення функції **f(x)** можна розрахувати в потрібних точках $\mathbf{x_i}$ з кроком **h**. При цьому **h** називають кроком чисельного диференціювання. Крок можна підібрати залежно від поведінки функції **f(x)** в околі точки **x**.

Запис формул через $\mathbf{y_i}$ відповідає постановці задачі диференціювання, коли функція **f(x)** задана таблицею, її аналітичний вигляд невідомий.

Чисельно значення похідної визначається заміною границі з диференціалом **dx** на відношення на інтервалі **h** у вигляді **розділеної різниці**. В залежності від кількості точок та напрямку зсуву **h** розрізняють праву різницю, ліву різницю та центральну різницю.

$$f'(x) \approx \frac{\Delta f(x)}{\Delta x} \approx \begin{cases} \dfrac{f(x+h) - f(x)}{h}, \text{права різниця} \\ \dfrac{f(x) - f(x-h)}{h}, \text{ліва різниця} \\ \dfrac{f(x+h) - f(x-h)}{2h}, \text{центральна різниця} \end{cases} \qquad h = dx.$$

Прості формули, за допомогою яких похідні в заданій точці можна наближено обчислити за кількома значеннями функції **f(x)** в заданій та близьких до неї точках забезпечують широке застосування чисельного підходу в комп'ютерні математиці.

При комп'ютерних розрахунках результат обчислення значення функції має обчислювальну похибку, мінімальне значення якої визначається похибкою округлення Δc.

$$\widetilde{f(x+h)} = \widetilde{f_{i+1}} = f(x+h) + \Delta_c(x+h) = f_{i+1} + \Delta_c,$$

$$\widetilde{f(x)} = \breve{f}_i = f(x) + \Delta_c(x) = f_i + \Delta_c,$$

$$\widetilde{f(x-h)} = \widetilde{f_{i-1}} = f(x-h) + \Delta_c(x-h) = f_{i-1} + \Delta_c,$$

де Δc – абсолютна гранична похибка округлення («комп'ютерний епсилон»).

Відповідно, розраховане значення похідної функції теж матиме похибку округлення Δ

$$\widetilde{f'(x)} = \breve{f'}_i \approx \begin{cases} \overrightarrow{\breve{f'}_i} = \dfrac{f_{i+1} + \Delta_c - f_i + \Delta_c}{h} = \dfrac{f_{i+1} - f_i}{h} + \dfrac{2\Delta_c}{h} \overrightarrow{= f'_i} + \Delta \\ \overleftarrow{\breve{f'}_i} = \dfrac{f_i + \Delta_c - f_{i-1} + \Delta_c}{h} = \overleftarrow{f'_i} + \Delta \\ \dot{\breve{f'}}_i = \dfrac{f_{i+1} + \Delta_c - f_{i-1} + \Delta_c}{2h} = \dot{f'}_i + \Delta \end{cases}.$$

Для всіх різновидів розділеної різниці ця похибка складає

$$\Delta = \frac{2\Delta_c}{h}.$$

Наявність обчислювальної похибки Δ робить операцію чисельного диференціювання **теоретично некоректною**. Для виконання умови визначення похідної необхідно зменшувати відстань h=dx→0. Одночасно зменшення призводить до зростання похибки округлення Δ.

Сумарна похибка диференціювання складається з похибки округлення та методичної похибки обчислення розділеної різниці

$$\Delta_\Sigma = \Delta + \Delta_M.$$

Оцінимо методичні похибки розділених різниць із застосуванням розкладання функції **`f(x)`** в ряд Тейлора:

$$f(x+h) = f(x) + \frac{f'(x)h}{1!} + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \cdots$$

$$f(x-h) = f(x) - \frac{f'(x)h}{1!} + \frac{f''(x)h^2}{2!} - \frac{f'''(x)h^3}{3!} + \cdots$$

Застосування наближення f(x+h) для правої різниці дає

$$\overrightarrow{f'_\iota} = \frac{f_{i+1} - f_i}{h} = \frac{f(x+h) - f(x)}{h} \approx \frac{f'(x)h}{h} + \frac{f''(x)h}{2}.$$

Застосування наближення **f(x-h)** для лівої різниці дає

$$\overleftarrow{f'_\iota} = \frac{f_i - f_{i-1}}{h} = \frac{f(x) - f(x-h)}{h} \approx f'(x) + \frac{f''(x)h}{2}.$$

Застосування наближення f(x+h) та f(x-h) для центральної різниці дає

$$\dot{f'}_\iota = \frac{f_{i+1} - f_{i-1}}{2h} = \frac{f(x+h) - f(x-h)}{2h} = \frac{2f'(x)h}{2h} + \frac{2f'''(x)h^3}{12h}$$
$$= f'(x) + \frac{f'''(x)h^2}{6}.$$

Методичні похибки правої та лівої різниць є однаковими та прямо пропорційні значенню добутку другої похідної функції та кроку диференціювання. Центральна різниця має меншу похибку, яка прямо пропорційна добутку третьої похідної функції та квадрата кроку диференціювання

Для чисельного диференціювання функції **f(x)** в околі точки **x** існує оптимальний крок диференціювання, який забезпечує мінімальну абсолютну похибку.

Оптимальний крок чисельного диференціювання можна визначити знаходженням екстремуму

$$\Delta'_\Sigma = 0 = (\Delta_M + \Delta)'.$$

Для лівої та правої різниць він становить

$$\left(\frac{f''(x)h}{2} + \frac{2\Delta_c}{h}\right)' = \frac{f''(x)}{2} - \frac{2\Delta_c}{h^2} = 0\,, \quad h = \sqrt{\frac{4\Delta_c}{f''(x)}}.$$

Для центральної різниці він становить

$$\left(\frac{f'''(x)h^2}{6} + \frac{2\Delta_c}{h}\right)' = \frac{2f'''(x)h}{6} - \frac{2\Delta_c}{h^2} = 0, \quad h = \sqrt[3]{\frac{6\Delta_c}{f'''(x)}}.$$

Диференціювання таблично визначених функцій

Для диференціювання функції **f(x),** що визначена на відрізку [x_0, x_N] множиною значень [y_0, y_N], її замінюють інтерполяційним поліномом **P(x)** з відповідною похибкою інтерполяції

$$f(x) = P_N(x) + \Delta_N(x).$$

Значення похідної отримують диференціюванням полінома **P(x)**

$$f^{(k)}(x) = P_N^{(k)}(x) + \Delta_N^{(k)}(x).$$

Для використання береться перший доданок

$$f^{(k)}(x) \approx P_N^{(k)}(x), \;\; 1 < k < N.$$

Чисельне диференціювання на основі інтерполяційних формул Ньютона.

Для розрахунку значення похідних в точці, що лежить на початку таблиці, застосовують першу інтерполяційну формулу Ньютона.

$$f(x) \approx y_0 + t\Delta y_0 + \frac{t(t-1)\Delta^{(2)} y_0}{2!} + \cdots + \frac{t(t-1)\ldots(t-N+1)}{N!}\Delta^{(N)} y_0\,, \qquad t = \frac{x - x0}{h}$$

$$N_N(x = x0 + th) = y_0 + \sum_{i=1}^{N} \frac{\prod_{k=0}^{i-1}(t-k)}{i!}\Delta^{(i)} y_0.$$

Для точки y_0 кінцеві різниці становлять

$$\Delta y_0 = y_1 - y_0,$$

$$\Delta^{(2)} y_0 = \Delta y_1 - \Delta y_0 = y_2 - y_1 - y_1 + y_0 = y_2 - 2y_1 + y_0,$$

$$\Delta^{(3)} y_0 = \Delta^{(2)} y_1 - \Delta^{(2)} y_0 = y_3 - 2y_2 + y_1 - y_2 + 2y_1 - y_0 = y_3 - 3y_2 + 3y_1 - y_0.$$

Добуток $\prod_{k=0}^{i-1}(t-k)$ може бути обчислено як суму

$$t(t-1)(t-2)\ldots(t-k) = t^{k+1} - S_k^1 t^k + S_k^2 t^{k-1} + \cdots + (-1)^k S_k^k t,$$

де S_k^i – сума добутків усіх натуральних чисел від **1** до **k** по **i** множниках

$S_1^1=1 \;\; S_2^1=1+2=3 \;\; S_2^2=1\cdot 2=2$

$S_3^1=1+2+3=6$ $S_3^2=1\cdot2+1\cdot3+2\cdot3=11$ $S_3^3=1\cdot2\cdot3=6$

$S_4^1=1+2+3+4=10$ $S_4^2=1\cdot2+1\cdot3+1\cdot4+2\cdot3+2\cdot4+3\cdot4=35$

$S_4^3=1\cdot2\cdot3+1\cdot2\cdot4+1\cdot3\cdot4+2\cdot3\cdot4=50$ $S_4^4=1\cdot2\cdot3\cdot4=24$

З урахуванням наведених вище виразів наближення функції можна записати як

$$f(x)\approx y_0+t\Delta y_0+\frac{t^2-t}{2}\Delta^{(2)}y_0+\frac{t^3-3t^2+2t}{6}\Delta^{(3)}y_0 + \frac{t^4-6t^3+11t^2-6t}{24}\Delta^{(4)}y_0+\cdots + \frac{t^N-S_{N-1}^1t^{N-1}+S_{N-1}^2t^{N-2}+\cdots+(-1)^{N-1}S_{N-1}^{N-1}t}{N!}\Delta^{(N)}y_0.$$

Похідна функції виглядатиме як

$$f'(x)\approx\frac{1}{h}\left(\Delta y_0+\frac{2t-1}{2!}\Delta^{(2)}y_0+\frac{3t^2-6t+2}{3!}\Delta^{(3)}y_0+\frac{4t^3-18t^2+22t-6}{4!}\Delta^{(4)}y_0+\frac{Nt^{N-1}-(N-1)S_{N-1}^1t^{N-2}+\cdots+(-1)^{N-1}S_{N-1}^{(N-1)}}{N!}\Delta^{(N)}y_0\right)=\frac{1}{h}\left(\Delta y_0+\sum_{i=2}^{N}\frac{it^{i-1}+\sum_{k=2}^{i}(-1)^{k-1}(i-k+1)t^{i-k}S_{i-1}^{k-1}}{i!}\Delta^{(i)}y_0\right). \qquad (8.1)$$

Для розрахунку значення похідних в точці, що лежить в кінці таблиці, застосовують другу інтерполяційною формулою Ньютона.

$$f(x)=N_N(x)=y_{\mathrm{N}}+\sum_{i=1}^{N}\frac{\Delta^{(i)}y_{\mathrm{N-i}}\prod_{k=0}^{i-1}(t+k)}{i!} =y_N+\Delta y_{N-1}t+\frac{\Delta^{(2)}y_{N-2}t(t+1)}{2}+\frac{\Delta^{(3)}y_{N-3}t(t+1)(t+2)}{6}+\cdots$$

Вираз для похідної якої виглядає як

$$f'(x)=\frac{1}{h}(\Delta y_{N-1}+\frac{2t+1}{2}\Delta^{(2)}y_{N-2}+\frac{3t^2+6t+2}{6}\Delta^{(3)}y_{N-3}+\cdots)\,. \qquad (8.1')$$

Формули чисельного диференціювання значно спрощуються, якщо значення похідних обчислюється у вузлах інтерполювання.

Для точки $x=x_0$ $(t=0)$ можна записати

$$f'(x_0)=y_0'=\frac{1}{h}\left(\Delta y_0-\frac{\Delta^{(2)}y_0}{2}+\frac{\Delta^{(3)}y_0}{3}-\frac{\Delta^{(4)}y_0}{4}+\cdots+\frac{(-1)^{N-1}\Delta^{(N)}y_0}{N}\right),$$

$$f'(x_i) = \frac{1}{h}\sum_{k=1}^{N-i}\frac{(-1)^{k-1}\Delta^{(k)}y_i}{k}\,. \qquad (8.2)$$

Чисельне диференціювання на основі інтерполяційних формул Лагранжа.

Похідна інтерполяційної формули Лагранжа у відносних координатах **x=x0+t·h** для рівновіддалених вузлів

$$L_N(x) = L(x_0 + th) = \sum_{i=0}^{N}\frac{(-1)^{N-i}y_i\prod_{k=0}^{N}(t-k)}{i!\,(N-i)!\,(t-i)}.$$

Визначається як

$$\frac{dL_N(x)}{dx} = [dx = hdt] = \frac{1}{h}\sum_{i=0}^{N}\frac{(-1)^{N-i}y_i}{i!(N-i)!}\frac{d}{dt}\left[\frac{\prod_{k=0}^{N}(t-k)}{t-i}\right]. \qquad (8.3)$$

Чисельне диференціювання на основі "ковзаючої" інтерполяції

На основі інтерполяційних формул **глобальної** інтерполяції отримують практичні залежності для диференціювання «ковзаючим» способом, коли до уваги беруться тільки найближчі до точки x вузли таблиці.

За точку $\mathbf{x_0}$ вибирають найближче табличне значення аргументу $\mathbf{x_i}$, яке менше за **x**. При цьому треба враховувати, що діапазон сітки знижується до **N-i**.

Формула Ньютона 1-го порядку (лінійне наближення) з використанням двох точок має вигляд

$$f(x) \approx y_0 + t\Delta y_0,$$

формула Ньютона 2-го порядку (квадратичне наближення) з використанням трьох точок має вигляд

$$f(x) \approx y_0 + t\Delta y_0 + t(t-1)\frac{\Delta^{(2)}y_0}{2}.$$

Значення похідної для наведених наближень становить

$$\frac{df(x)}{dx} = \frac{d(x0+th)}{hdt} = \begin{cases} \dfrac{\Delta y_0}{h}, \text{лінійне наближення} \\ \dfrac{\Delta y_0}{h} + (2t-1)\dfrac{\Delta^{(2)}y_0}{2h}, \text{квадратичне наближення} \end{cases}$$

Для вузла **x0** по трьох точках похідну можна обчислити як:

$$f'(x0) = \frac{\Delta y_0}{h} - \frac{\Delta^{(2)} y_0}{2h} = \frac{y_1 - y_0}{h} - \frac{\Delta y_1 - \Delta y_0}{2h}$$
$$= \frac{2y_1 - 2y_0 - (y_2 - y_1 - y_1 + y_0)}{2h} = \frac{-y_2 + 4y_1 - 3y_0}{2h}.$$

Для двох вузлів (**N=1**).

$$f'(x) \approx \frac{1}{h}\left(\frac{(-1)^{1-0} y_0}{0!(1-0)!}\left(\frac{t(t-1)}{t}\right)' + \frac{(-1)^{1-1} y_1}{1!(1-1)!}\left(\frac{t(t-1)}{t-1}\right)'\right) = \frac{1}{h}\left(\frac{-y_0}{1} + y_1\right). \quad (8.4)$$

Для трьох вузлів (N = 2).

$$f'(x) \approx \frac{1}{h}\left(\frac{(-1)^{2-0} y_0}{0!(2-0)!}\left(\frac{t(t-1)(t-2)}{t}\right)' + \frac{(-1)^{2-1} y_1}{1!(2-1)!}\left(\frac{t(t-1)(t-2)}{t-1}\right)' + \right.$$
$$\left. \frac{(-1)^{2-2} y_2}{2!(2-2)!}\left(\frac{t(t-1)(t-2)}{t-2}\right)'\right) = \frac{1}{h}\left(\frac{(2t-3)y_0}{2} - (2t-2)y_1 + \frac{(2t-1)y_2}{2}\right). \quad (8.5)$$

Для вузла $\mathbf{x_0}$, коли **t=0**, по трьох точках::

$$f'(x_0) = \frac{-3y_0 + 4y_1 - y_2}{2h}.$$

Вираз збігається зі значенням, яке обчислюється за формулами Ньютона, що підтверджує еквівалентність застосування інтерполяції Лагранжа та інтерполяції Ньютона з математичної точки зору.

На практиці диференціювання на базі «ковзаючої» інтерполяції забезпечує значне скорочення обчислювальних витрат, особливо для сітчастих функцій великої розмірності.

Проведемо порівняння результатів формул чисельного диференціювання для знаходження похідної першого порядку логарифмічної функції **f(x)=ln(x)** для сітки від 1 до 2 з кроком 0.2 у вузлах сітки та для x=1.12.

Результати розрахунків за формулою Ньютона (8.2), Лагранжа (8.3), «ковзаючою» інтерполяцією по двох точках (8.4) та трьох точках (8.5) наведено в табл. 8.1.

З наведених результатів очевидно, що найвищу точність забезпечує чисельне диференціювання за Лагранжем. Не дивлячись на математичну тотожність інтерполяції за Ньютоном та Лагранжем, похибка чисельного

диференціювання за Ньютоном є більшою. Диференціювання за «ковзаючою» інтерполяцією має не тільки суттєво більшу похибку, але й вимагає додаткових алгоритмічних заходів для того, щоб проводити розрахунки в кінці таблиці.

Таблиця 8.1 – Результати чисельного диференціювання

x	Ньютон	Лагранж	2 точки	3 точки	Точно
1.0	0.9575	0.9991	0.9116	0.9820	1.000
1.2	0.8335	0.8335	0.7708	0.8223	0.8333
1.4	0.7372	0.7142	0.6677	0.7070	0.7143
1.6	0.6637	0.6251	0.5889	0.6200	0.6250
1.8	0.6137	0.5554	0.5268	-	0.5556
2.0	0.5938	0.5005	-	-	0.5000
1.12	0.8793	0.8931	0.9116	0.8223	0.8929

8.1. Засоби диференціювання СКМ MathCAD, Matlab

В MathCAD диференціювання проводиться в один клік введення звичних символів з панелі розширених операторів.

Для чисельного диференціювання в призначені функції **diff**, **gradient**, **polyder**.

Функція **y=diff(x <,n,dim>)** розраховує вектор/матрицю **y** кінцевих різниць між елементами вектора/матриці **x**. Для аргумента-матриці результатом є матриця з елементами **Y=[X(2,:)-X(1,:); X(3,:)-X(2,:); ... X(p,:)-X(p-1,:)]**.

Опціональний аргумент **n** визначає кількість разів застосування функції. Наприклад, **diff(X,3)** є тотожнім **diff(diff(diff(X)))**.

Опціональний цілочисельний аргумент **dim** визначає в разі матриці x напрям дії. Значення 1 розраховує різницю між елементами стовпців, 2 – рядків.

Наприклад,

```
X = [1 1 2 3 5]; Y = diff(X); Z=diff(X,2);
Y =  0     1     1     2
Z = 1 0 1
X = [1 1 1; 5 5 5; 25 25 25]; Y = diff(X)
Y = 4     4     4
    20    20    20
X = [1 3 5;7 11 13;17 19 23]; Y = diff(X,1,2)
Y =  2     2
     4     2
     2     4
```

Функція

<[> FX <,FY,FZ,...,FN]>=gradient(F <,hx,hy,...,hN>)

розраховує першу розділену різницю вектора-рядка/матриці **F**.

Для аргументу **F** у вигляді вектора-рядка результатом є аналог першої похідної dF/dx або одномірного градієнта у вигляді вектора-рядка. Опціональний аргумент **hx** визначає крок диференціювання. За замовчанням крок дорівнює 1. Тобто результати дій функцій **gradient** та **diff** є подібними. Різниця полягає в розмірі результату. Довжина вектора-результату функції **diff** є на 1 меншою від довжини вектора-аргументу. Довжина вектора-результату функції **gradient** дорівнює довжині вектора-аргументу.

```
>>diff([1 2 3 4 5])      ans=1   1    1    1
>>gradient([1 2 3 4 5]) ans=1   1    1    1    1
```

Для аргументу **F** у вигляді матриці/тензора результатом є компоненти двомірного/багатомірного градієнта у вигляді матриці. Елементи **FX**, **FY**, … результуючого вектора є дискретними аналогами частинних похідних $\partial F/\partial x$, $\partial F/\partial y$, … відповідно. Для **FX** дія проводиться по стовпцях **F**, для **FY** – по рядках **F**.

Опціональні аргументи **hx, hy** визначають крок диференціювання. За замовчанням крок дорівнює 1.

Для елементів **F**, крім першого та останнього розраховується «центральна» розділена різниця: **G(:,j)=(F(:,j+1)-F(:,j-1))/(2h**).

Для першого елементу **F** розраховується «права однобічна» розділена різниця: **G(:,1)=(F(:,2)-F(:,1))/h**.

Для останнього елементу **F** розраховується «ліва однобічна» розділена різниця: **G(:,N)=(F(:,N)-F(:,N-1))/h**.

```
[a b]=gradient([1 2 3; 9 8 7; 4 6 5])
a=1    1     1       b=  8    6      4
  -1   -1    -1          1.5    2      1
  2    0.5   -1           -5   -2     -2
```

Розглянемо використання функції **gradient** для розрахунку градієнта функції $f(x,y) = x^2 \cdot y^2$ в точці (-0.5 0).

Аналітичним значенням градієнта є $\nabla f = 2x \cdot y^2\vec{\imath} + x^2 2y\vec{\jmath} = 0\vec{\imath} + 0\vec{\jmath}$.

Чисельний розрахунок в СКМ Matlab може мати наступний вигляд:

```
x = -1:0.5:0; y = x';
f = x.^2.*y.^2;
[fx,fy]=gradient(f,0.5);
x0 = -0.5;y0 = 0;
fx= -1.5    -1     -0.5
    -0.375  -0.25  -0.125
     0       0      0
fy= -1.5   -0.375  0
    -1     -0.25   0
    -0.5   -0.125  0
indt = find((x == x0) & (y == y0)); indt= 6
f_grad = [fx(indt) fy(indt)]; f_grad = 0 -0.125
```

Результат відрізняється від теоретичного та є невірним. Причина полягає в визначеному діапазоні змінних. Граничне значення векторів задано **x,y** є 0, що співпадає з точкою визначення градієнта. Якщо розширити діапазон так, щоб точка оцінювання потрпила всередину діапазону, то результат чисельного обчислення співпаде з теоретичним:

```
x = -1:0.5:0.5; y = x';
```

```
[fx,fy]=gradient(f,0.5);
fx= -1.5    -1         0       0.5
    -0.375  -0.25      0       0.125
     0       0         0       0
    -0.375  -0.25      0       0.125
fy= -1.5    -0.375     0      -0.375
    -1      -0.25      0      -0.25
     0       0         0       0
     0.5    0.125      0       0.125
indt = find((x == x0) & (y == y0)); indt= 7
f_grad = [fx(indt) fy(indt)]; f_grad = 0 0
```

Функція

<[> q <,d]> = polyder(a <,b>) розраховує похідну від поліноміальної функції. Обов'язковий аргумент **a** є вектором-рядком, який містить коефіцієнти полінома. Опціональний аргумент **b** є також вектором-рядком, який містить коефіцієнти полінома.

В разі використання однієї змінної-результату розраховується похідна полінома **a** або похідна від добутку поліномів **a·b.** Результат є вектором, елементами якого є коефіцієнти результуючого полінома.

В разі використання двох змінних **q, d** в векторі-результаті розраховується похідна від ділення поліномів **a/b.** Результуючий вектор містить **q** коефіцієнти діленого, результуючий вектор **q** – коефіцієнти дільника.

Наприклад, для поліномів $a(x) = 3x^5 - 2x^3 + x + 5$ та $b(x) = x^2 - 10x + 15$ **a=[3 0 -2 0 1 5]; b=[1 -10 15];**

рядок **polyder(a)** розраховує похідну від поліному a(x):

ans =15 0 -6 0 1 $\frac{da(x)}{dx} = 15x^4 - 6x^2 + 1,$

рядок **polyder(a,b)** розраховує похідну від добутку поліномів a(x)b(x):

ans =21 -180 215 80 -87 -10 -35

$$\frac{da(x)b(x)}{dx} = 21x^6 - 180x^5 + 215x^4 + 80x^3 - 87x^2 - 10x - 35,$$

рядок **[d q]=polyder(a,b)** розраховує похідну від частки поліномів a(x)/b(x):

```
d =9  -120   223    40   -91   -10    65
q =1   -20   130  -300   225
```

$$\frac{da(x)/b(x)}{dx} = \frac{9x^6-120x^5+223x^4+40x^3-91x^2-10x+65}{x^4-20x^3+130x^2-300x+225}$$

Контрольні запитання

В яких випадках проводиться чисельне диференціювання?

Назвіть типи розділених різниць?

Яку величину має мінімальна похибка округлення для чисельного диференціювання?

В чому полягає теоретична некоректність дії чисельного диференціювання?

Як обирають базову точку для «ковзаючого» диференціювання?

Яким чином розраховують похідну в ?

Яку розмірність має результат функції **diff**?

В якому разі результтати функцій **gradient** та **diff** співпадають?

Для чого застосовується функція **polyder**?

ДОДАТОК А

```
function varargout = myinterp(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
             'gui_OpeningFcn', @myinterp_OpeningFcn, ...
             'gui_OutputFcn',  @myinterp_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code


function myinterp_OpeningFcn(hObject, eventdata, ...
handles, varargin)
 handles.output = hObject;
 handles.q=inline('sin(x)');
 handles.x=[0 0.5 1 1.5]; handles.n=1000;handles.point=0.3;
 handles.y=handles.q(handles.x);
 guidata(hObject, handles);
 myinit(hObject, eventdata, handles);


function pushbutton1_Callback(hObject, eventdata, handles)
d=handles.q(handles.point);
 tic;
 for i=1:handles.n
   tmp=interp1(handles.x,handles.y,handles.point,'linear');
 end
 t=toc;
 set(handles.blintimetext,'String',t);
 set(handles.blinrestext,'String',num2str(tmp,4));
 set(handles.blinerrortext,'String',...
 num2str(abs(tmp-d),4));
 tic;
 for i=1:handles.n
    tmp=interp1(handles.x,handles.y,handles.point,...
   'cubic');
```

```
end
t=toc;
set(handles.bqtimetext,'String',t);
set(handles.bqrestext,'String',num2str(tmp,4));
set(handles.bqerrortext,'String',...
num2str(abs(tmp-d),4));
tic;
for i=1:handles.n
   tmp=handles.y(1)+(handles.y(2)-handles.y(1))...
   *(handles.point-handles.x(1))/(handles.x(2)-...
   handles.x(1));
end
t=toc;
set(handles.lintimetext,'String',t);
set(handles.linrestext,'String',num2str(tmp,4));
set(handles.linerrortext,'String',...
num2str(abs(tmp-d),4));
tic;
for i=1:handles.n
   tmp=quadinterp(handles.x,handles.y,handles.point);
end
t=toc;
set(handles.partimetext,'String',t);
set(handles.parrestext,'String',num2str(tmp,4));
set(handles.quaderrortext,'String',...
num2str(abs(tmp-d),4));
function Fedit_Callback(hObject, eventdata, handles)
  handles.q=inline(get(hObject,'String'));
  guidata(hObject,handles);
  myinit(hObject, eventdata, handles);

function x1edit_Callback(hObject, eventdata, handles)
 handles.x(1)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function X2edit_Callback(hObject, eventdata, handles)
 handles.x(2)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles)';

function X3edit_Callback(hObject, eventdata, handles)
 handles.x(3)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles);
```

```
function Xedit_Callback(hObject, eventdata, handles)
 handles.point=str2num(get(hObject,'String'));
 guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function Nedit_Callback(hObject, eventdata, handles)
 handles.n=str2int(get(hObject,'String'));
 guidata(hObject,handles);

function myinit(hObject, eventdata, handles)
 handles.y=handles.q(handles.x);
 guidata(hObject,handles);
 set(handles.realtext,'string',...
 num2str(handles.q(handles.point),4));
 set(handles.y1text,'string',...
 num2str(handles.q(handles.x(1)),4));
 set(handles.y2text,'string',...
 num2str(handles.q(handles.x(2)),4));
 set(handles.y3text,'string',...
 num2str(handles.q(handles.x(3)),4));
 set(handles.y4text,'string',...
 num2str(handles.q(handles.x(4)),4));

function X4edit_Callback(hObject, eventdata, handles)
 handles.x(4)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x);
 guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function z=quadinterp(x,y,a)
 d=(x(1)-x(2))*(x(2)-x(3))*(x(3)-x(1));
 CC(1)=(y(1)*(x(3)-x(2))+y(2)*(x(1)-x(3))+...
 y(3)*(x(2)-x(1)))/d;
 CC(2)=(y(1)*(x(2)^2-x(3)^2)+y(2)*(x(3)^2-...
 x(1)^2)+y(3)*(x(1)^2-x(2)^2))/d;
 CC(3)=(y(1)*(x(3)-x(2))*x(3)*x(2)+y(2)*(x(1)-...
 x(3))*x(1)*x(3)+y(3)*(x(2)-x(1))*x(2)*x(1))/d;
 z=CC(1)*a^2+CC(2)*a+CC(3);
```

Рисунок А.1 - Лістинг інтерполяційної моделі []

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Томашевський, В. М. Моделювання систем: Підручник / В. М. Томашевський. – К. : Видавнича група BHV, 2005. – 352 с.
2. Амосов, А. А. Вычислительные методы для инженеров: Учебное пособие / А. А. Амосов, Ю. А. Дубинский, Н. В. Копченкова. – М. : Высш. школа, 1994. – 544 с.
3. Советов, Б. Я. Моделирование систем: Учеб. для вузов — 3-е изд., перераб. и доп. / Б. Я. Советов, С. А. Яковлев. — М. : Высш. шк., 2001. — 343 с.
4. Шеннон, Р. Имитационное моделирование – искусство и наука / Р. Шеннон. – М. : Мир, 1978. - 212 с.
5. Струтинський, В. Б. Математичнс моделювання процесів та систем механіки: Підручник / В. Б. Струтинський. – Житомир : ЖІТІ, 2001. - 612 с.
6. Лазарєв, Ю. Ф. Моделювання на ЕОМ. Навчальний посібник / Ю. Ф. Лазарєв. – К. : Політехніка, 2007. - 290 с.
7. Фельдман, Л. П. Чисельні методи в інформатиці. Підручник / Л. П. Фельдман. – К. : Видавнича груп BHV, 2006. - 480с.
8. Гаврилюк І. П. Методи обчислень: Підручник: У 2 ч. / І. П. Гаврилюк. – К. : Вища школа, 1995. – 367 с.
9. Бахвалов, Н. С. Численные методы [Текст] : учеб. пособие для физ.-мат. специальностей вузов / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков ; под общ. ред. Н. И. Тихонова. – 6-е изд. – М. : БИНОМ, 2008. — 636 с.
10. Кравченко, І. В. Інформаційні технології: Системи комп'ютерної математики [Електронний ресурс]: навч. посіб. для студ. спеціальності «Автоматизація та комп'ютерно-інтегровані технології» / І. В. Кравченко, В. І. Микитенко; КПІ ім. Ігоря Сікорського. - Електронні текстові дані (1 файл: 5,57 Мбайт). - Київ : КПІ ім. Ігоря Сікорського, 2018. – 243 с.
11. Кетков, Ю. Л. MATLAB 7: программирование, численные методы / Ю. Л. Кетков, А. Ю. Кетков, М. М. Шульц. – СПб. : БХВ-Петербург, 2005. – 752 с
12. Кравченко, І. В. Моделювання оптико-електронних приладів: Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп'ютерно-інтегровані технології»

/ І. В. Кравченко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,35 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 113 с.
13. Гилат, А. MATLAB. Теория и практика. 5-е изд. / Амос Гилат, Пер. с англ. Смоленцев Н. К. – М. : ДМК Пресс, 2016. – 416 с.
14. Макаров, Е. Г. Инженерные расчеты в MathCAD. Учебный курс / Е. Г. Макаров. – СПб. : Питер, 2005. – 448 с.
15. Трусов П. В. Введение в математическое моделирование: Учебное пособие / П. В. Трусов. – М. : Логос, 2005. – 440 с.
16. ДСТУ 2226-93. Автоматизовані системи. Терміни та визначення. - Чинний з 1994-07-01. – ДП «УкрНДНЦ», 1996. – 93 с.
17. ДСТУ 2938-94 (ISO 2382-1:1993). Системи оброблення інформації. Основні поняття. Терміни та визначення. Чинний з 1996-01-01. – ДП «УкрНДНЦ», 1996. - 36 с.
18. ДСТУ ISO/IEC 2382:2017 (ISO/IEC 2382:2015, IDT) Інформаційні технології. СЛОВНИК ТЕРМІНІВ. - Чинний від 2019–01–01. - ДП «УкрНДНЦ», 2020. – 468 с.
19. Кельтон, В Имитационное моделирование. Классика. 3-е изд. / В. Кельтон, А. Лоу. – СПб. : Питер; Киев: Издательская группа BHV, 2004. – 847 с.
20. Возняк, Л. С. Чисельні методи: Методичний посібник для судентів природничих спеціальностей / Л. С. Возняк, С. В. Шарин. – Івано-Франківськ : Плай, 2001.- 64 с.
21. Ляшенко, Б. М. Методи обчислень: навчально-методичний посібник для студентів фізико-математичного факультету / Б. М. Ляшенко, О. М. Кривонос, Т. А. Вакалюк. – Житомир : Вид-во ЖДУ, 2014. - 228 с.
22. Задачин, В. М. Чисельні методи / В. М. Задачин. - Х. : ХНЕУ, 2014. – 180 с.
23. Банди, Б. Методы оптимизации. Вводный курс / Б. Банди. -М. : Радио и связь, 1988. – 128 с.
24. Моделювання інформаційно-вимірювальних систем: Конспект лекцій для студентів спеціальності 152 «Метрологія та інформаціно-вимірювальна техніка» [Електронне видання] / Укл. І. В. Кравченко. – К. : НТУУ «КПІ», 2017. – 79 с.