Комп'ютерне моделювання систем та процесів. Методи обчислень

Частина 2

Вінниця
ВНТУ
2013

Міністерство освіти і науки, молоді та спорту України
Вінницький національний технічний університет

Комп'ютерне моделювання систем та процесів

Методи обчислень

Частина 2

Навчальний посібник

Вінниця
ВНТУ
2013

УДК 519.876.5(075)
ББК 32.97в6я73
К32

Автори:
Р. Н. Квєтний, І. В. Богач, О. Р. Бойко, О. Ю. Софина, О. М. Шушура

Рекомендовано Міністерством освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом підготовки «Системна інженерія». Лист № 1/11-1254 від 01.02.2012 р.

Рецензенти:
Б. П. Русин, доктор технічних наук, професор
Г. С. Фінін, доктор фізико-математичних наук, ст.н.сп.
А. М. Пєтух, доктор технічних наук, професор

К32 **Комп'ютерне** моделювання систем та процесів. Методи обчислень. Частина 2 : навчальний посібник / [Р. Н. Квєтний, І. В. Богач, О. Р. Бойко та інші]; за заг. ред. Р. Н. Квєтного. – Вінниця : ВНТУ, 2013. – 235 с.

ISBN 978-966-641-521-2 (частина 2)

Друга частина навчального посібника, в якому розглянуто найпоширеніші чисельні методи, що зустрічаються в типових інженерних та наукових задачах, методи оптимізації та основи математичного моделювання, а також методи цифрової обробки сигналів та зображень, фрактальний та інтервальний аналіз. Призначено для студентів напряму підготовки «Системна інженерія» при вивченні дисципліни «Комп'ютерне моделювання систем та процесів», але може бути використано при вивченні широкого спектру дисциплін цього та інших напрямів, які пов'язані з комп'ютерними обчисленнями та обробкою даних, сигналів, зображень, а також для наукової роботи студентів, аспірантів, інженерів та вчених. Наведено широкий спектр прикладів та задач.

УДК 519.876.5(075)
ББК 32.97в6я73

ISBN 978-966-641-519-9 (загальний)
ISBN 978-966-641-521-2 (частина 2)

ЗМІСТ

Вступ 6

РОЗДІЛ 1 ЦИФРОВА ОБРОБКА СИГНАЛІВ 8

1.1 Загальні відомості та поняття 8
1.2 Загальна структура системи цифрової обробки аналогових сигналів 11
1.3 Дискретні та неперервні сигнали 13
1.4 Теорема Котельникова..... 16
1.5 Дискретні перетворення сигналів 18
1.5.1 Спектр Фур'є неперервних та дискретних сигналів 19
1.5.2 Дискретне перетворення Фур'є 21
1.5.3 Застосування ДПФ 26
1.5.4 Ортогональні перетворення в діадних базисах 28
1.6 Згортка. Кореляція..... 29
1.7 Цифрова фільтрація сигналів 32
Контрольні завдання та запитання 37

РОЗДІЛ 2 МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ 39

2.1 Класичні методи обробки зображень 39
2.1.1 Математичні моделі зображень 39
2.1.2 Статистичні методи аналізу зображень 42
2.1.3 Фільтрація зображень 44
2.1.3.1 Оптимальна лінійна фільтрація 46
2.1.3.2 Нелінійна фільтрація 48
2.1.3.3 Інверсні фільтри в задачах обробки зображень ... 51
2.1.4 Методи на основі динамічних моделей 54
2.1.5 Методи на основі декомпозиції на власні вектори 57
2.1.6 Методи класифікації елементів зображень 58
2.1.7 Методи визначення контурів елементів зображень та сегментації 62
2.2 Фрактальні методи 63
2.3 Вейвлет-перетворення 76
2.4 Нейронні мережі в задачах обробки зображень 84
Контрольні завдання та запитання 87

РОЗДІЛ 3 ІНТЕРВАЛЬНИЙ АНАЛІЗ 89

3.1 Класична інтервальна арифметика 89
3.2 Інтервальне розширення та звуження 91
3.3 Диференціювання та інтегрування в інтервальному аналізі 93
3.4 Інтервальні методи розв'язання диференціальних рівнянь..... 96
3.4.1 Інтервальний метод другого порядку для розв'язання звичайних диференціальних рівнянь 97
3.4.2 Інтервальні методи типу Рунге-Кутта 98
3.4.3 Метод Крукеберга 99
3.5 Подання інтервальної функції через граничні дійсні функції 100
3.6 Розширення інтервальної арифметики 103
Контрольні завдання та запитання 105

РОЗДІЛ 4 МЕТОДИ ОПТИМІЗАЦІЇ І ПЛАНУВАННЯ 107

4.1 Класична постановка задачі оптимізації 107
4.2 Класифікація задач оптимізації 108
4.3 Багатокритеріальна оптимізація 109
4.4 Гладка оптимізація 110
4.4.1 Умови Куна-Таккера 111
4.4.2 Чисельні методи гладкої оптимізації 111
4.4.3 Методи зведення загальної задачі оптимізації до задачі без обмежень..... 115
4.5 Опукла оптимізація 117
4.5.1 Простий субградієнтний метод опуклої оптимізації..... 118
4.5.2 Методи розтягу простору..... 119
4.6 Негладка оптимізація за методом координатного спуску 119
4.7 Стохастична оптимізація 120
4.8 Лінійне програмування 120
4.8.1 Симплекс-метод 121
4.8.2 Транспортна задача..... 121
4.8.3 Цілочислове лінійне програмування..... 122
4.8.4 Загальна задача лінійної оптимізації..... 123
4.9 Теорія ігор 124
4.10 Динамічне програмування 125
4.11 Варіаційні задачі 126
4.12 Системна оптимізація 127

4.13 Застосування теорії графів до розв'язання оптимізаційних задач 129
4.14 Застосування активних експериментів при ідентифікації моделей 131
Контрольні запитання та завдання 141

ДОДАТОК А
Чисельний розрахунок деяких задач 144

ДОДАТОК Б
Приклади використання пакета MathCad для розв'язання чисельних задач 164

ДОДАТОК В
Приклади використання пакета MATLAB для розв'язання чисельних задач 197

ДОДАТОК Г
Стислий англійсько-російсько-українсько-польський словник технічних термінів (Concise English-Russian-Ukrainian-Polish Dictionary of Terms)..... 225

Література..... 232

ВСТУП

Одним з головних напрямків науково-технічного прогресу протягом вже кількох десятиріч є розвиток методів і засобів інформатики та обчислювальної техніки. Використання методів математичного моделювання та комп'ютерного розв'язання інженерних і наукових задач дозволяє значно підвищити ефективність процесів проектування та управління. Впровадження персональних комп'ютерів, комп'ютерних інформаційних мереж, побудова та розвиток INTERNET, широке та різноманітне використання методів математичного моделювання привели до розширення як практичної, так і теоретичної баз комп'ютерної математики. Математичне комп'ютерне моделювання стало головним засобом дослідження складних процесів і систем, на якому базуються сучасні підходи до проектування, оптимізації та управління в різних галузях науки і техніки. Обчислювальна математика стала основою для реалізації та комп'ютерного розрахунку методів математичного моделювання.

Метою цієї книги, що видана в двох частинах, є ознайомлення широкого кола студентів, науковців, інженерно-технічних працівників з основними поняттями комп'ютерного моделювання систем і процесів та методами розв'язання на комп'ютерах сучасних задач обчислювальної математики, що виникають в процесі дослідження й проектування систем управління та автоматики.

Посібник призначений для студентів напрямів підготовки «Системна інженерія» та «Комп'ютерна інженерія», але також може бути використаний для інших спеціальностей при вивченні дисциплін, пов'язаних з чисельними методами, методами обробки даних, оптимізацією та плануванням експерименту.

Теоретичною основою посібника стали відомі роботи в області моделювання та обчислювальної математики Л. Коллатця, Дж. Форсайта, Р. Мура, Д. Кнута, А. Самарського, Б. Демідовича, І. Марона, В. Скурихіна, М. Бусленка , А. Крилова, А. Верланя, Г. Марчука та інших вітчизняних і закордонних вчених. В посібнику узагальнено досвід багаторічного викладання Р. Н. Квєтним курсів з обчислювальної математики та моделювання, використано результати наукових і навчально-методичних розробок співавторів цієї книги.

Перший розділ першої частини книги присвячений розгляду основних понять і підходів до побудови математичних комп'ютерних моделей. В наступних розділах розглянуто найбільш поширені задачі обчислювальної математики, методи та алгоритми їх розв'язання. Ці задачі розглянуто в першій частині навчального посібника. Друга частина присвячена задачам обробки сигналів та зображень, що дуже поширені в практиці комп'ютерного моделювання, інтервальним методам та ще деяким

проблемам, які пов'язані з розрахунками в процесі математичного моделювання й можуть бути корисними для сучасних науковців та інженерів.

Використання посібника передбачає знання основ програмування, а алгоритмізація та реалізація комп'ютерних програм може здійснюватись з застосуванням будь-якої зручної для користувача мови програмування. В книзі наведено приклади розв'язання деяких задач з використанням процедур програмних систем MathCad, MATLAB та інших, але не дається загальних основ використання цих систем, що не входить до задач цього посібника. Автори намагалися зробити так, щоб викладений матеріал не дуже підпадав під вплив часу, тому робили наголос на висвітленні базових понять. Особливу увагу приділено алгоритмізації обчислювальних процедур як основ застосування комп'ютерної математики.

Розділ 1 першої частини підготовлено Р. Н. Квєтним, О. Ю. Софиною, О. М. Шушурою; розділи 2, 3 – Р. Н. Квєтним, І. В. Богач, О. М. Шушурою; розділи 4, 5 – Р. Н. Квєтним; розділ 6 – Р. Н. Квєтним та О. М. Шушурою. В другій частині розділи 1, 2 – О. Ю. Софиною (2.2 – разом з І. В. Богач); розділ 3 – О. Р. Бойко; розділ 4 – Р. Н. Квєтним. Загальну редакцію книги здійснено Р. Н. Квєтним.

РОЗДІЛ 1 ЦИФРОВА ОБРОБКА СИГНАЛІВ

1.1 Загальні відомості та поняття

Цифрова обробка сигналів (ЦОС) (digital signal processing) – це область обчислювальної техніки, що динамічно розвивається та охоплює як технічні, так і програмні засоби. Спорідненими областями для цифрової обробки сигналів є теорія інформації, зокрема, теорія оптимального прийому сигналів і теорія розпізнавання образів. При цьому в першому випадку основним завданням є виділення сигналу на фоні шумів і завад різної фізичної природи, а в другому – автоматичне розпізнавання, тобто класифікація та ідентифікація сигналу.

При цифровій обробці використовується подання сигналів у вигляді послідовностей чисел або символів. Ціль такої обробки може полягати в оцінюванні характерних параметрів сигналу або в перетворенні сигналу у форму, що, в деякому змісті, більше зручна. Такі формули класичного чисельного аналізу, як формули для інтерполяції, інтегрування й диференціювання, безумовно, є алгоритмами цифрової обробки. Наявність швидкодійних цифрових ЕОМ сприяє розвитку все більше складних і раціональних алгоритмів обробки сигналів; останні ж успіхи в технології інтегральних схем обіцяють високу економічність побудови дуже складних систем цифрової обробки сигналів. Цифрова обробка сигналів застосовується в таких різних галузях, як біомедицина, акустика, звукова локація, радіолокація, сейсмологія, зв'язок, системи передачі даних, ядерна техніка і багатьох інших.

Цифрова обробка сигналів є альтернативою традиційній аналоговій. До її найважливіших якісних переваг відносять можливість реалізації будь-яких як завгодно складних (оптимальних) алгоритмів обробки з гарантованою і незалежною від дестабілізувальних факторів точністю; програмованість та функціональна гнучкість; можливість адаптації до сигналів, що обробляються; технологічність.

Розвиток нової точки зору на цифрову обробку сигналів було прискорено відкриттям в 1965 р. ефективних алгоритмів для обчислень перетворень Фур'є. Цей клас алгоритмів став відомий як швидке перетворення Фур'є (ШПФ, fast Fourier transform). Можливості ШПФ були значними з декількох точок зору. Багато алгоритмів обробки сигналів, отриманих на цифрових ЕОМ, вимагали часу обробки на декілька порядків більшого, ніж реальний час. Часто це було пов'язане з тим, що спектральний аналіз був важливою складовою частиною обробки сигналів, а ефективні засоби для його виконання не були відомі. Алгоритм швидкого перетворення Фур'є зменшив час обчислення перетворення Фур'є на кілька порядків. Це дозволило створити дуже складні алгоритми обробки

сигналів у реальному часі. Крім того, з урахуванням можливостей дійсної реалізації алгоритму швидкого перетворення Фур'є на спеціалізованому цифровому пристрої, багато алгоритмів обробки сигналів, що були раніше непрактичними, стали знаходити втілення на спеціалізованих пристроях.

Методами ЦОС є математичні співвідношення або алгоритми, відповідно до яких виконуються обчислювальні операції над цифровими сигналами. До них належать алгоритми цифрової фільтрації, спектрально-кореляційного аналізу, модуляції та демодуляції сигналів, адаптивної обробки та ін.

Засобами реалізації ЦОС є жорстка логіка, програмовані логічні інтегральні схеми, мікропроцесори загального призначення, мікроконтролери, персональні комп'ютери (комп'ютерна обробка сигналів) та цифрові сигнальні процесори.

У технічних галузях знань термін «сигнал» (signal, від латинського signum – знак) часто використовується в широкому діапазоні значень, без дотримання строгої термінології. Під ним розуміють і технічний засіб (матеріальний носій) для передачі, обігу і використання інформації – електричний, магнітний, оптичний сигнал; і фізичний процес, що являє собою матеріальне відображення інформаційного повідомлення – зміна певного параметра носія інформації (напруги, частоти, потужності електромагнітних коливань, інтенсивності світлового потоку тощо) у часі, у просторі або залежно від зміни значень будь-яких інших аргументів (незалежних змінних).

Всі ці поняття можна об'єднати одним технічним терміном: сигнал – це фізична величина, що містить у собі певні відомості про певний об'єкт або процес.

Термін «сигнал» дуже часто ототожнюють із поняттями «дані» (data) і «інформація» (information). Дійсно, ці поняття взаємозалежні і не існують одне без іншого, але належать до різних категорій.

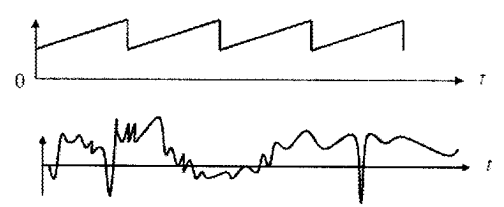
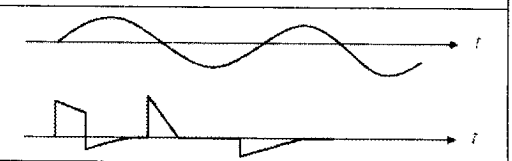
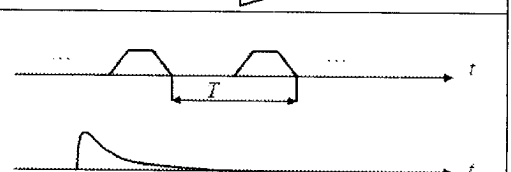
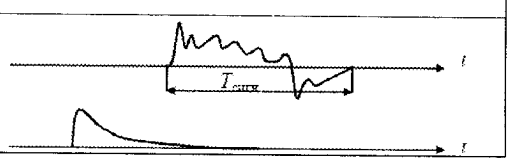
Термін «signal» у світовій практиці є загальноприйнятим для характеристики форми подання даних, при якій дані розглядаються у вигляді послідовності значень скалярних величин (аналогових, числових, графічних та ін.) залежно від зміни будь-яких змінних значень (часу, енергії, температури, просторових координат та ін.). З урахуванням цього, надалі під терміном «сигнал» у точному значенні цього слова будемо розуміти певним чином впорядковане *відображення* певних даних про характер зміни у просторі і часі або за будь-якою іншою зміною фізичних величин, фізичних властивостей або фізичного стану об'єкта досліджень. Оскільки дані про вимірювання містять інформацію як про основні цільові параметри об'єкта досліджень, так і про різні супутні фактори впливу, то в широкому розумінні цього слова можна вважати, що сигнал є відображенням загальної вимірювальної інформації. При цьому матеріальна форма носіїв сигналів, так само як і форма їхнього

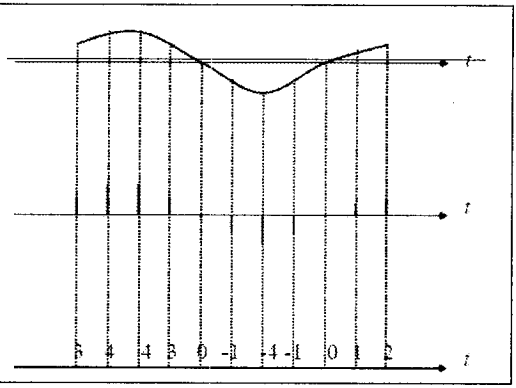
відображення в будь-якому фізичному процесі, значення не має.

Отже, сигнал – це інформаційна функція, що несе повідомлення про фізичні властивості, стан або поведінку будь-якої фізичної системи, об'єкта або середовища, а метою обробки сигналів у найзагальнішому змісті можна вважати отримання певних інформаційних відомостей, які відображені в цих сигналах (корисна або цільова інформація), і перетворення цих відомостей у форму, зручну для сприйняття і подальшого використання.

Множина сигналів надзвичайно різноманітна. Проте, вибираючи певні критерії розбіжності, можна навести приблизну класифікацію сигналів (таблиця 1.1). Однак слід пам'ятати, що у реальному житті сигнали часто не вкладаються в рамки чистої класифікації. Наприклад, будь-який реальний детермінований сигнал має випадкову шумову складову.

Таблиця 1.1 – Класифікація сигналів

	Тип (клас) сигналів	Геометричне зображення
1.	а) **Детерміновані** (значення s (t) відомо в будь-який момент часу t) б) **Випадкові** (передбачити точне значення s (t) неможливо)	
2.	а) **Неперервні** (без розривів першого роду) б) **Імпульсні** (з розривами першого роду)	
3.	а) **Періодичні** (період T) б) **Неперіодичні** $T = \infty$	
4.	а) **Кінцевої довжини** ($T_{сиг}$) б) **Нескінченної довжини** $T_{сиг} = \infty$	

5.	а) **Аналогові** (існують у будь-який момент часу t і можуть приймати будь-яке значення в інтервалі $[S_{\min}, S_{\max}]$) б) **Дискретні** (існують тільки в дискретні моменти t_k, тобто є послідовністю імпульсних відліків) в) **Цифрові** (послідовність цифрових відліків)	

1.2 Загальна структура системи цифрової обробки аналогових сигналів

Системи ЦОС безпосередньо оперують із послідовностями цифрових кодів, які називають цифровими сигналами. Такі сигнали обробляються процесором ЦОС, що є операційним або обчислювальним ядром системи. Алгоритмічна обробка аналогових сигналів цифровими засобами припускає їхнє попереднє перетворення в цифрову форму, а в системах з аналоговим виходом – із цифрової форми в аналогову. Загальній структурній схемі системи цифрової обробки аналогових сигналів (рис. 1.1) відповідає ланцюжок функціональних перетворень сигналу виду: $A/A \Rightarrow A/Ц \Rightarrow Ц/Ц \Rightarrow Ц/A \Rightarrow A/A$ («аналог/аналог», «аналог/цифра», «цифра/цифра», «цифра/аналог», «аналог/аналог»), реалізованих відповідно аналоговим фільтром нижніх частот ФНЧ1, аналого-цифровим перетворювачем АЦП, процесором ЦОС, цифроаналоговим перетворювачем ЦАП і аналоговим фільтром нижніх частот ФНЧ2.

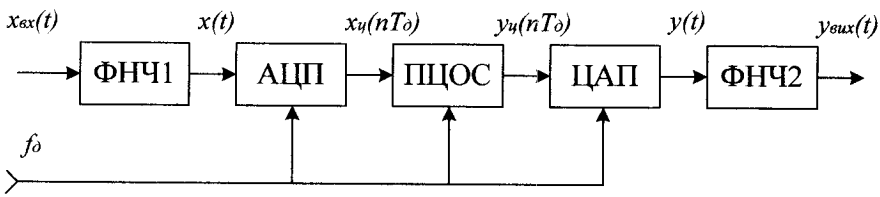


Рисунок 1.1 – Загальна структура системи ЦОС

Вхідний сигнал системи ЦОС $x_{вх}(t)$ надходить на АЦП через аналоговий фільтр нижніх частот ФНЧ1 із частотою зрізу $\omega_з$. Фільтр

обмежує смугу частот вхідного сигналу (охоплюючи й супутні йому шуми та завади) максимальною частотою $\omega_m \approx \omega_з$, що задовольняє умову $\omega_m < \omega_D / 2$, де $\omega_D = 2\pi f_D$ – частота дискретизації сигналу. Він послабляє спотворення накладання при дискретизації сигналів з необмеженим за частотою спектром і називається протимаскувальним.

Аналого-цифрове перетворення (analog-to-digital conversion) охоплює дискретизацію сигналу за часом, квантування за рівнем і цифрове кодування (рис. 1.2)

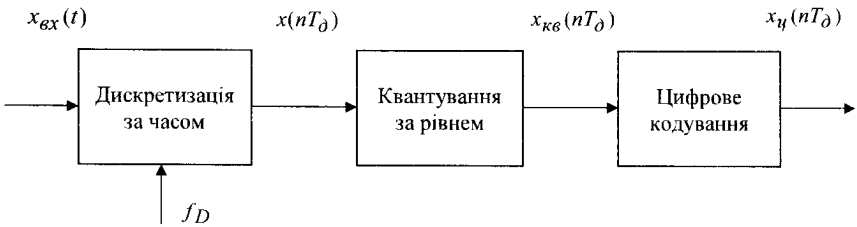


Рисунок 1.2 – Послідовність операцій аналого-цифрового перетворення сигналу

У результаті утворюється дискретний сигнал $x(nT_д)$, що відповідає вибіркам аналогового сигналу *x(t)* у дискретні рівновіддалені моменти часу $nT_д$, дискретний квантованный сигнал $x_{кв}(nT_д)$, що відрізняється кінцевою множиною прийнятих ним значень, і цифровий сигнал $x_ц(nT_д)$ у вигляді послідовності цифрових двійкових кодів із числом розрядів, що відповідають розрядності АЦП. Процесором ЦОС, відповідно до заданого алгоритму цифрової обробки (оператором Ф), вхідний цифровий сигнал $x_ц(nT_д)$ перетвориться у вихідний цифровий сигнал системи $y_ц(nT_д) = Ф[x_ц(nT_д)]$.

Аналоговий вихідний сигнал системи $y_{вих}(t)$ виходить із цифрового сигналу $y_ц(nT_д)$ за допомогою ЦАП, що перетворює його у квантований за рівнем аналоговий сигнал ступінчастої форми, і аналогового ФНЧ2, яким обмежується частотний спектр і заглушаються високочастотні компоненти вихідного сигналу. Цей фільтр із частотою зрізу $\omega_з < \omega_D / 2$ називають також згладжувальним.

Дискретизація за часом, або просто дискретизація (digitization) являє собою процедуру взяття миттєвих значень – відліків аналогового сигналу *x(t)* з інтервалом часу, рівним періоду дискретизації $T_д$. Приклад дискретизованого сигналу наведено на рис. 1.3.

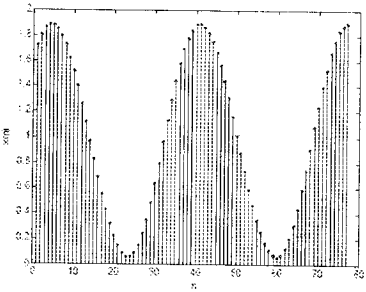


Рисунок 1.3 – Дискретизований сигнал

Квантування за рівнем (квантування – quantization) здійснюється з метою подання точних значень відліків $x_ц(nT_Д)$ у вигляді двійкових чисел кінцевої розрядності – квантованих відліків $x_ц(nT_Д)$. Для цього динамічний діапазон дискретного сигналу $x(nT_Д)$ розбивається на кінцеве число дискретних рівнів – рівнів квантування – і кожному відліку за певним правилом привласнюється значення одного з найближчих рівнів, між якими він виявляється. Рівні квантування кодуються двійковими числами розрядності b, що залежить від числа рівнів квантування R

$$R \le 2^b.$$

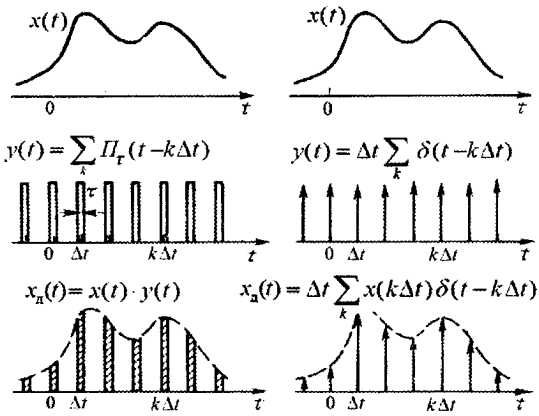
Сукупність квантованных відліків $x_ц(nT_Д)$, $n = 0, 1, 2, \ldots$ називають *цифровим сигналом* (digital signal).

Сукупність елементів ФНЧ1, АЦП, ЦАП і ФНЧ2 системи цифрової обробки аналогових сигналів, що виконують перетворення сигналів виду А/А, А/Ц і Ц/А, утворює підсистему її аналогового введення-виведення або аналого-цифровий інтерфейс.

1.3 Дискретні та неперервні сигнали

Більшість реальних сигналів (наприклад, звукових) є неперервними функціями. Для цифрової обробки таких сигналів їх потрібно перевести в цифрову форму. Один із способів зробити це – рівномірно за часом виміряти значення сигналу на певному проміжку часу і ввести отримані значення амплітуд. Якщо робити вимірювання досить часто, то за значеннями отриманого дискретного сигналу можна буде досить точно відновити вигляд вихідного неперервного сигналу.

Дискретні сигнали (discrete signal) $x_D(t)$ утворюють шляхом множення аналогового сигналу $x(t)$ на так звану функцію дискретизації $y(t)$, яка являє собою періодичну послідовність коротких імпульсів, що слідують з кроком дискретизації Δt (рис. 1.4, а). В ідеальному випадку як функція дискретизації використовується періодична послідовність дельта-функцій (рис. 1.4, б).



а) б)

Рисунок 1.4 – Дискретизація сигналу

Процес вимірювання величини сигналу через рівні проміжки часу називається рівномірною (за часом) дискретизацією. Багато пристроїв для введення даних здійснюють дискретизацію.

Наприклад, звукова карта дискретизує сигнал з мікрофона, сканер дискретизує сигнал, що надходить з фотоелемента. У результаті дискретизації безперервний (аналоговий) сигнал перетворюється у послідовність значень. Пристрій, що виконує цей процес, називається аналого-цифровим перетворювачем (АЦП, analogue-to-digital converter, ADC). Частота, з якою АЦП здійснює вимірювання значень аналогового сигналу і видає його цифрові значення, називається частотою дискретизації.

Інтервал $T = k\Delta t$ називають періодом дискретизації, частотою дискретизації є обернена величина

$$f_n = \frac{1}{t}.$$

Значення послідовності в моменти часу nT називають відліками. Дискретний сигнал може бути як дійсним, так і комплексним. В

останньому випадку його дійсна та уявна частини описуються дійсними послідовностями

$$x(nT) = x_1(nT) + jx_2(nT).$$

Математично дискретний сигнал визначають:

– функцією дискретного часу nT_∂ : $x(nT_\partial) = x(t)\big|t = nT_\partial$, $n = 0, 1, 2, \ldots$ що відповідає вибіркам аналогового сигналу в дискретні періодично повторювані моменти часу;

– функцією номера вибірки $n : x(n) = x(nT_\partial)\big|T_\partial = 1$, що в загальному випадку не пов'язана з часом;

– функцією неперервного часу t:

$$x(t) = x(t) f_\delta(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_\partial) = \sum_{n=-\infty}^{\infty} x(nT_\partial)\delta(t - nT_\partial),$$

що її отримують множенням аналогового сигналу $x(t)$ на функцію дискретизації $f_\delta(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_\partial)$ у вигляді періодичної послідовності 8 імпульсів з періодом, що є рівним:

$$\delta(t - nT_\partial) = \begin{cases} \infty, t = nT_\partial \\ 0, t \neq nT_\partial \end{cases}.$$

Графічно дискретні сигнали представляються функцією номера вибірки n або дискретного часу nT_∂ (рис. 1.5).

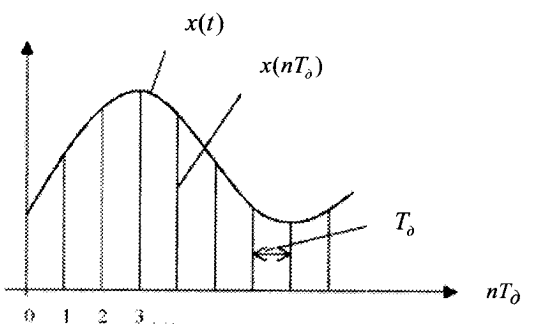


Рисунок 1.5 – Графік неперервного $x(t)$ та дискретного $x(nT_\partial)$ сигналу

1.4 Теорема Котельникова.

Для того, щоб відновити вихідний неперервний сигнал з дискретизованого з малими похибками, необхідно раціонально вибрати крок дискретизації. Тому при перетворенні аналогового сигналу в дискретний обов'язково виникає питання про величину кроку дискретизації. Якщо аналоговий сигнал має низькочастотний спектр, обмежений деякою верхньою частотою $F_в$ (тобто функція *u(t)* має вигляд кривої, яка плавно змінюється, без різких змін амплітуди), то навряд чи на деякому невеликому часовому інтервалі дискретизації Δt ця функція може істотно змінюватися за амплітудою.

Очевидно, що точність відновлення аналогового сигналу за послідовністю його відліків залежить від величини інтервалу дискретизації. Чим він коротше, тим менше буде відрізнятися функція *u(t)* від плавної кривої, що проходить через точки відліків. Однак зі зменшенням інтервалу дискретизації, істотно зростає складність і обсяг обчислень. При досить великому інтервалі дискретизації Δt зростає ймовірність спотворення або втрати інформації при відновленні аналогового сигналу.

Оптимальна величина інтервалу дискретизації встановлюється **теоремою Котельникова**, яка має важливе теоретичне та практичне значення: дає можливість правильно здійснити дискретизацію аналогового сигналу та визначає оптимальний спосіб його відновлення на приймальному кінці за відліковим значенням.

Відповідно до однієї з найбільш відомих і простих інтерпретацій теореми Котельникова довільний сигнал *s(t)*, спектр якого обмежений деякою частотою $F_в$, може бути повністю відновлений за послідовністю своїх відлікових значень, що слідують з інтервалом часу

$$\Delta t = \frac{1}{2F_в}.$$

Інтервал дискретизації Δt та частоту $F_в$ часто називають інтервалом і частотою Найквіста. Аналітично теорема Котельникова подається рядом

$$s(t) = \sum_{k=-\infty}^{\infty} s(k\Delta t) \frac{\sin \frac{\pi}{\Delta t}(t - k\Delta t)}{\frac{\pi}{\Delta t}(t - k\Delta t)},$$

де k – номер відліку; $s(k\Delta t)$ – значення сигналу в точках відліку.

Фізичний зміст цієї теореми стає зрозумілим, якщо розглянути спектри сигналів $S(t)$ і $S_D(t)$.

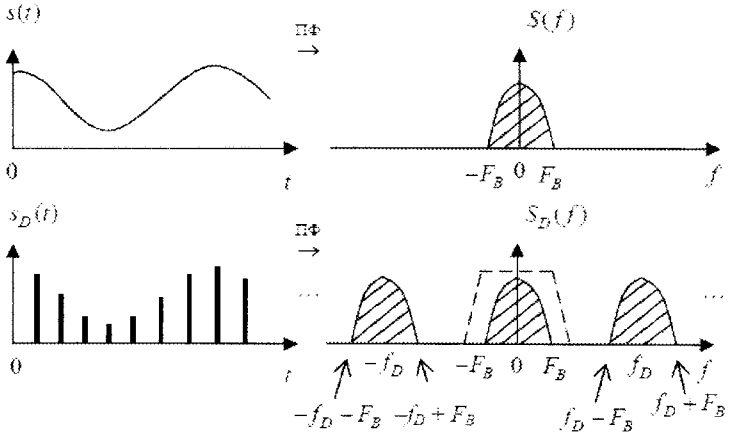


Рисунок 1.6 – Теорема Котельникова

З рис. 1.6 видно, що $S_D(f)$ містить у собі $S(f)$ ще й нескінченне число копій $S(f)$, зсунутих одна відносно одної на частоту дискретизації f_D. Якщо пропустити сигнал $S_D(f)$ через фільтр нижніх частот (ФНЧ), амплітудно-частотна характеристика якого показана на цьому ж рисунку, на виході ФНЧ залишиться тільки $S(f)$, тобто відновиться вихідний сигнал *s(t)*.

При $f_D > 2F_в$ копії не перетинаються з основним пелюстком спектра $S_D(f)$ і таке відновлення можливе.

При $f_D = 2F_в$ копії стикаються з основним пелюстком, однак виділення вихідного сигналу *s(t)* ще можливе за допомогою ідеального ФНЧ із нескінченною крутизною спаду амплітудно-частотної характеристики (АЧХ).

При $f_D < 2F_в$ пелюстки спектра $S_D(f)$ перекриваються і відновлення вихідного сигналу *s(t)* неможливим.

На практиці частоту f_D завжди вибирають більшою, ніж $2F_в$, тому що будь-який фільтр має далеко не нескінченну крутизну спаду АЧХ.

Спектр реального сигналу рідко має точну верхню границю $F_в$. Найчастіше $S(f)$ зменшується з ростом частоти асимптотично наближуючись до нуля. У такому випадку на в поміщають ФНЧ, що має частоту, рівну вихідного аналогового сигналу. Його призн

спектра за межами $F_в$ і тим самим уникнути перекриття пелюстків спектра $S_D(f)$.

На практиці ця теорема має величезне значення. Наприклад, відомо, що більшість звукових сигналів можна з деякою мірою точності вважати сигналами з обмеженим спектром. Їх спектр, в основному, лежить нижче 20 кГц. Це означає, що при дискретизації з частотою не менш 40 кГц, ми можемо достатньо точно відновити вихідний аналоговий звуковий сигнал за його цифровими відліками. Абсолютної точності досягти не вдасться, тому що в природі не буває сигналів з ідеально обмеженим спектром.

Пристрій, який інтерполює дискретний сигнал до безперервного, називається цифроаналоговим перетворювачем (ЦАП, digital-to-analogue converter). Ці пристрої застосовуються, наприклад, в програвачах компакт-дисків для відтворення звуку з цифрового звукового сигналу, записаного на компакт-диск. Частота дискретизації звукового сигналу під час запису на компакт-диск становить 44100 Гц. Таким чином, і ЦАП на CD-плеєрі працює на частоті 44100 Гц.

1.5 Дискретні перетворення сигналів

Крім звичного подання сигналів і функцій у вигляді залежності їх значень від певних аргументів (часу, лінійної або просторової координати тощо) при аналізі й обробці даних широко використовується математичний опис сигналів за аргументами. Можливість такого опису визначається тим, що будь-який як завгодно складний за своєю формою сигнал, що не має нескінченних значень на своєму інтервалі, можна подати у вигляді суми більше простих сигналів, і, зокрема, у вигляді суми найпростіших гармонічних коливань, що виконується за допомогою перетворення Фур'є. Відповідно, математично розкладання сигналу на гармонічні складові описується функціями значень амплітуд і початкових фаз коливань за неперервним або дискретним аргументом. Сукупність амплітуд гармонічних коливань розкладання називають амплітудним спектром сигналу, а сукупність початкових фаз – фазовим спектром. Обидва спектри разом утворюють повний частотний спектр сигналу, що за точністю математичного подання тотожний динамічній формі опису сигналу.

Крім гармонічного ряду Фур'є застосовуються й інші види розкладання сигналів: за функціями Уолша, Адамара, Вейвлета та інших, крім того, існують розкладання за поліномами Чебишова, Лаггера, Лежандра та інших. У ЦОС широко використовується дискретне перетворення Фур'є (ДПФ, discrete Fourier transform) і алгоритм його швидкого обчислення – швидке перетворення Фур'є (ШПФ). Вони дозволяють адекватно описувати в частотних координатах всі, крім наймиттєвіших (< 1 с), сигнали; зрізані за частотою Фур'є-компоненти описують дані більш правдоподібно, ніж будь-які інші степеневі ряди.

1.5.1 Спектр Фур'є неперервних та дискретних сигналів

Нехай $x(t)$ – неперервний сигнал, що задовольняє умову $\int_{-\infty}^{\infty}|x(t)|dt<\infty$. Сигнал $x(t)$ у цьому випадку може бути поданий у вигляді інтегрального розкладу за системою комплексних синусоїдальних функцій – інтеграла Фур'є:

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}X(\omega)e^{j\omega t}d\omega=\int_{-\infty}^{\infty}X(f)e^{j2\pi ft}df\,, \qquad (1.1)$$

де $X(\omega)$ – комплексна функція, що визначає амплітуду та фазову затримку комплексної синусоїди із частотою ω: $e^{i\omega t}=\cos(\omega t)+i\sin(\omega t)$. У загальному випадку ця функція визначена на всій осі частот $\omega\in[-\infty,\infty]$ і називається вона Фур'є-спектром сигналу $x(t)$.

У свою чергу Фур'є-спектр $X(\omega)$ може бути отриманий з вихідного сигналу $x(t)$ за допомогою співвідношення:

$$X(\omega)=\int_{-\omega}^{\infty}x(t)e^{-j\omega t}dt. \qquad (1.2)$$

Співвідношення (1.1), (1.2) являють собою пари інтегральних перетворень Фур'є, причому (1.2) – пряме перетворення Фур'є, (1.1) – обернене перетворення Фур'є.

Відмітимо, що сигнал $x(t)$ и Фур'є-спектр $X(\omega)$ – дві взаємно-однозначні характеристики, перша є часовим поданням сигналу, друга – частотним. Часове подання більш наочне та звичне для повсякденного сприйняття, друге – менш наочне, але винятково корисне при математичному описі перетворень сигналів у лінійних системах з постійними параметрами.

Основні властивості Фур'є-спектра $X(\omega)$:

1. Функція $X(\omega)$ в загальному випадку є комплексною:

$$X(\omega)=\mathrm{Re}\,X(\omega)+i\,\mathrm{Im}\,X(\omega)=|X(\omega)|e^{i\arg X(\omega)}=A(\omega)e^{i\Phi(\omega)}.$$

Функцію $A(\omega)=|X(\omega)|$ називають амплітудним спектром (іноді магнітудою спектра), вона визначає дійсну амплітуду синусоїди із частотою ω, що бере участь у формуванні сигналу. Функцію

$\Phi(\omega) = \arg X(\omega) = \arctan\left(\frac{\operatorname{Im} X(\omega)}{\operatorname{Re} X(\omega)}\right)$ називають фазовим спектром, вона показує фазовий зсув, якому варто піддати комплексну синусоїду частоти ω перед підсумовуванням при відновленні вихідного сигналу.

2. Внаслідок дійсності сигналу $x(t)$ функція $X(\omega)$ має комплексно-спряжену симетрію

$$X(\omega) = X^*(-\omega),$$
$$\operatorname{Re} X(\omega) = \operatorname{Re} X(-\omega), \qquad \operatorname{Im} X(\omega) = -\operatorname{Im} X(-\omega),$$
$$|X(\omega)| = |X(-\omega)|, \qquad Arg X(\omega) = -Arg X(-\omega);$$

3. Енергія спектра Фур'є обмежена й дорівнює енергії вихідного сигналу (рівність Парсеваля):

$$\frac{1}{(2\pi)^2}\int_{-\infty}^{\infty} X^2(\omega)d\omega = \int_{-\infty}^{\infty} x^2(t)dt < \infty.$$

У теорії неперервних лінійних систем з постійними параметрами широко використовується поняття перетворення Лапласа (s-перетворення)

$$X(s) = \int_{-\omega}^{\infty} x(t)e^{-st}dt \tag{1.3}$$

функції, визначеної на комплексній s-площині: $s = \upsilon + j\omega$.

При цьому пряме перетворення Фур'є (1.2) може розглядатися як перетворення Лапласа, обчислене на уявній осі в s-площині:

$$X(\omega) = X(s = j\omega).$$

У зв'язку із цим, у літературі часто можна зустріти позначення для Фур'є-спектра – $X(j\omega)$, в якому є вказівка на те, що це спектр саме неперервного сигналу.

В теорії дискретних лінійних систем замість s-перетворення Лапласа широко використовується поняття Z-перетворення дискретного сигналу

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}. \tag{1.4}$$

Z-перетворення має сенс для тих значень комплексної змінної z, при

яких ряд (1.4) збігається.

Z-перетворення лінійне, завдяки чому воно успішно використовується при описі лінійних дискретних систем. Вихідна послідовність може бути відновлена за допомогою оберненого Z-перетворення

$$x(n)=\frac{1}{2\pi j}\oint_C X(z)z^{n-1}dz,$$

де C – замкнутий контур, що охоплює все особливі точки функції $X(z)z^{n-1}$.

Спектр Фур'є дискретних сигналів. Спектром Фур'є послідовності $x(n)$ називають комплексну функцію $X(e^{j\omega})$

$$X(e^{j\omega})=\sum_{n=-\infty}^{\infty}x(n)e^{-j\omega n}, \qquad (1.5)$$

$$x(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi}X(e^{j\omega})e^{j\omega n}d\omega. \qquad (1.6)$$

Вираз (1.6) показує, як вихідна послідовність може бути зібрана з дискретизованих комплексних синусоїд різних частот, узятих з вагами $X(e^{j\omega})$. Порівняння (1.5) з (1.4) показує, що спектр Фур'є $X(e^{j\omega})$ є просто Z-перетворенням, обчисленим на одиничному колі $Z=e^{j\omega}$ в комплексній Z-площині. Властивості спектра Фур'є дискретних сигналів подібні до властивостей спектра Фур'є неперервних сигналів. Однак є принципова відмінність. Спектр $X(e^{j\omega})$ періодичний за частотою з періодом 2π. Тому його значення розглядають на одному періоді – або $[-\pi,\ \pi]$ або $[0,\ 2\pi]$.

1.5.2 Дискретне перетворення Фур'є

Дискретне перетворення Фур'є (ДПФ) є базовим алгоритмом цифрової обробки сигналів у частотній області. Завдяки наявності ефективних алгоритмів його обчислення – алгоритмів швидкого перетворення Фур'є (ШПФ) – ДПФ широко використовується для цілей цифрової фільтрації та спектрально-кореляційного аналізу сигналів.

Для сигналу, заданого у вигляді дискретної послідовності $S(n)$, пряме й обернене дискретне перетворення Фур'є (ДПФ) мають вигляд

$$S(k)=\sum_{n=0}^{N-1}S(n)\exp\left[-j\frac{2\pi nk}{N}\right],\quad k=\overline{0,N-1}; \qquad (1.7)$$

$$S(n)=\frac{1}{N}\sum_{k=0}^{N-1}S(k)\exp\left[j\frac{2\pi nk}{N}\right],\quad n=\overline{0,N-1},\tag{1.8}$$

де k – номер гармоніки із частотою f_k, N – обсяг вибірки. $S(k)$, визначений як комплексний спектр сигналу, можна подати у вигляді

$$S(k)=A(k)-jB(k)=C(k)e^{-j\varphi(k)},\tag{1.9}$$

де амплітудно-частотна (АЧХ) і фазочастотна (ФЧХ) характеристики сигналу відповідно визначаються

$$C(k)=\sqrt{A^2(k)+B^2(k)};\tag{1.10}$$

$$\varphi(k)=\operatorname{arctg}\frac{B(k)}{A(k)}+2\pi m.\tag{1.11}$$

Обернене ДПФ можна також виконати за допомогою співвідношення

$$S(n)=\frac{1}{N}\sum_{k=0}^{N-1}C(k)\cos\left[\varphi(k)+\frac{2\pi}{N}kn\right],\quad n=\overline{0,N-1}.\tag{1.12}$$

Виходячи з (1.12), оцінювання форми сигналу при використанні тільки інформації про його ФЧХ можна зробити за допомогою формули

$$\hat{S}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\cos\left(\varphi(k)+\frac{2\pi}{N}\cdot n\cdot k\right).\tag{1.13}$$

Якість відновлення сигналів можна поліпшити, додатково використовуючи в (1.13) різні вагові функції, наприклад, трикутну, експонентну й ін. Тоді вираз (1.13) можна переписати в такій формі

$$\hat{S}(n)=\frac{1}{N}\sum_{k=0}^{N-1}W(k)\cdot\cos\left(\varphi(k)+\frac{2\pi}{N}\cdot n\cdot k\right),\tag{1.14}$$

де $W(k)$ – прийнята вагова функція.

Похибку, що виникає при відновленні сигналу, можна оцінити за середнім значенням квадрата похибки:

$$\varepsilon^2 = \frac{\sum_{n=0}^{N-1} (\hat{S}(n) - S(n))^2}{\sum_{n=0}^{N-1} S^2(n)}.$$

Іншою відомою формою запису ДПФ є рівняння:

$$F(\Delta\xi) = \sum_{k=0}^{N-1} x(k\Delta x)\delta(x - k\Delta * x)e^{-j2\pi k\Delta\xi\Delta x} = \sum_{k=0}^{N-1} x_k e^{-j2\pi\frac{2x_{\max} * k}{N} * \frac{1}{2x_{\max}} n} =$$

$$= k * \sum_{k=0}^{N-1} x_k * e^{-j\frac{2\pi}{N}kn},$$

де $\begin{cases} \frac{2x_{\max}}{N} = \Delta x \\ \frac{1}{2x_{\max}} = \Delta\xi \end{cases}$.

Тоді легко отримати, що

$$\Delta x * \Delta\zeta = \frac{1}{N}.$$

Подібним способом можна отримати і для оберненого перетворення

$$x_m = k\sum F_n * e^{j\frac{2\pi}{N}nm},$$

$$k = \frac{1}{\sqrt{N}}.$$

Таким чином, в матричній формі:

$$F = \frac{1}{\sqrt{N}} E_N X,$$

$$E_N = \begin{vmatrix} W^0 & W^0 & W^0 & \dots & W^0 \\ W^0 & W^1 & W^2 & \dots & W^{N-1} \\ W^0 & W^2 & W^4 & \dots & W^{2(N-1)} \\ W^0 & W^3 & W^6 & \dots & W^{3(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ W^0 & W^{N-1} & W^{2(N-1)} & \dots & W^{(N-1)(N-1)} \end{vmatrix},$$

де $W^k = e^{-j2\pi k/N}$, а сама матриця ядра ДПФ називається матрицею дискретних експоненціальних функцій (ДЕФ, discrete exponential function). При цьому рядки матриці визначають набір ортогональних функцій або базис розкладання.

При виконанні перетворення Фур'є рядки матриці ядра задають набір ортогональних функцій, за якими виконується розкладання вихідного сигналу. Кожний елемент вектора результату визначає внесок відповідної ортогональної функції у формування вихідного сигналу. Для перетворення Фур'є, як і для будь-якого ортогонального перетворення, визначник матриці ядра перетворення E_N відмінний від "0", що дозволяє виконати як пряме, так і обернене перетворення

$$\begin{cases} F = \frac{1}{\sqrt{N}} E_N X \\ X = \frac{1}{\sqrt{N}} E_N^{-1} F \end{cases},$$

оскільки $\frac{1}{N} E_N * E_N^{-1} = \frac{1}{N} I_N = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{vmatrix}$.

Обчислення перетворень Фур'є вимагає дуже великого числа множень (приблизно N^2) і обчислень синусів. Існує спосіб виконати ці перетворення значно швидше: приблизно за $N \cdot \log_2 N$ операцій множення. Цей спосіб називається *швидким перетворенням Фур'є*. Алгоритм ШПФ – це спосіб швидкого обчислення ДПФ $X(jk) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn}$, що дозволяє усунути притаманну ДПФ надмірність. Дані перетворення ґрунтуються на властивостях комплексної експоненти $e^{-j(2\pi/N)kn}$, для зручності позначають W_N^{kn} ($W_N^{kn} = e^{-j(2\pi/N)kn}$), її симетрії $W_N^{(N-k)n} = W_N^{(N-n)k} = (W_N^{kn})^*$ і періодичності $W_N^{(N+k)(N+n)} = W_N^{kn}$ з періодом, рівним довжині оброблюваної реалізації сигналу N (числу точок ШПФ). Відповідно до останньої властивості експоненти $W_N^{pkn} = W_{N/p}^{kn}$ відповідає період N/p, де p – цілі числа, на які ділиться N. Використання даних властивостей в алгоритмах ШПФ дозволяє уникнути великого числа повторюваних при обчисленні ДПФ операцій.

У результаті швидкодія ШПФ може залежно від N в сотні разів перевершувати швидкодію стандартного алгоритму. При цьому слід підкреслити, що алгоритм ШПФ є точним. Він навіть точніше стандартного, тому що, скорочуючи число операцій, він веде до менших помилок округлення.

Однак у більшості алгоритмів ШПФ є особливість: вони здатні працювати лише тоді, коли довжина аналізованого сигналу N є степенем двійки. Зазвичай, це не є великою проблемою, оскільки аналізований сигнал завжди можна доповнити нулями до необхідного розміру. Число N називається розміром або довжиною ШПФ.

Для зображень, що являють собою двовимірний сигнал, спектром є також двовимірний сигнал. Базисні функції перетворення Фур'є мають вигляд добутків

$$h^{\sin}_{k_1,k_2}(n_1,n_2) = \sin\frac{2\pi k_1 n_1}{N_1}\cdot\sin\frac{2\pi k_2 n_2}{N_2},$$

$$h^{\cos}_{k_1,k_2}(n_1,n_2) = \sin\frac{2\pi k_1 n_1}{N_1}\cdot\sin\frac{2\pi k_2 n_2}{N_2},$$

де $N_1\times N_2$ розмір вихідного сигналу, він же – розмір спектра, k_1,k_2 – це номери базисних функцій (номери коефіцієнтів двовимірного ДПФ, при яких ці функції знаходяться). Оскільки розмір спектра дорівнює розміру вихідного сигналу,то $k_1 = 0,...,N_1-1; k_2 = 0,...,N_2-1$, n_1,n_2 – змінні аргументи базисних функцій. Оскільки область визначення базисних функцій збігається з областю визначення сигналу, то $n_1 = 0,...,N_1-1; n_2 = 0,...,N_2-1$.

Двовимірне ДПФ визначається нижченаведеними формулами

$$X[k_1,k_2] = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x[n_1,n_2]\cdot e^{-jn_1k_1(2\pi/N_1)}e^{-jn_2k_2(2\pi/N_2)};$$

$$x[n_1,n_2] = \frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X[k_1,k_2]\cdot e^{-jn_1k_1(2\pi/N_1)}e^{-jn_2k_2(2\pi/N_2)},$$

де $x[n_1,n_2]$ – вихідний сигнал, $X[k_1,k_2]$ – спектр вихідного сигналу.

Безпосереднє обчислення двовимірного ДПФ за наведеними формулами вимагає величезних обчислювальних витрат. Однак можна довести, що двовимірне ДПФ має властивість сепарабельності, тобто його можна обчислити окремо за двома вимірами. Для обчислення двовимірного ДПФ достатньо обчислити одновимірні комплексні ДПФ всіх рядків зображення, а потім обчислити в підсумковому «зображенні»

одновимірні комплексні ДПФ всіх стовпців. При цьому результати всіх одновимірних комплексних ДПФ потрібно записувати на місце вихідних даних для цих ДПФ. Наприклад, при обчисленні одновимірного ДПФ першого рядка зображення потрібно результат ДПФ записати в перший рядок цього зображення (він має той же розмір). Для цього потрібно кожний піксел зберігати у вигляді комплексного числа.

Таким чином, ефективний алгоритм обчислення ДПФ зображення полягає в обчисленні одновимірних ШПФ спочатку від всіх рядків, а потім – від всіх стовпців зображення.

1.5.3 Застосування ДПФ

Часто ДПФ застосовується для дослідження та аналізу спектра сигналу. При цьому, зазвичай, найбільш цікавими є лише амплітуди C_k окремих гармонік, а не їх фази. У цьому випадку спектр найчастіше відображається у вигляді графіка залежності амплітуди від частоти. Часто шкала амплітуд градується в децибелах. Децибели вимірюють не самі амплітуди, а їх відношення. Наприклад, різниця на 20 дБ означає відмінність амплітуд у 10 разів, різниця на 40 дБ означає відношення амплітуд у 100 разів. Відмінності амплітуд в 2 рази відповідає різниця приблизно в 6 дБ. Шкала частот також часто градується в логарифмічному масштабі.

Перед обчисленням спектра сигналу потрібно вибрати відрізок сигналу, на якому буде обчислюватися спектр. Довжина відрізка повинна бути степенем двійки (для роботи ШПФ), інакше сигнал треба доповнити нулями до потрібної довжини. Після цього до обраної ділянки сигналу застосовують ШПФ.

Коефіцієнти амплітуд розраховують за формулою

$$C_k = \sqrt{A_k^2 + B_k^2}\,.$$

При обчисленні спектра зазначеним методом можливий небажаний ефект. При розкладанні функції в ряд Фур'є вважається, що функція періодична, з періодом, рівним розміру ШПФ. Обчислюється спектр саме такої функції, а не тієї, сегмент якої використовується. При цьому на границях періодів така функція, напевно, буде мати розриви (адже вихідна функція не була періодичною). А розриви у функції суттєво впливають на її спектр, спотворюючи його.

Для усунення цього ефекту застосовуються так звані *вагові вікна*. Вони плавно зменшують значення функції поблизу границь ділянки, що аналізується. Вагові вікна мають форму, схожу на гауссіан. Обрану для

аналізу ділянку сигналу множать на вагове вікно, яке усуває розриви функції при «зацикленні» даної ділянки сигналу. «Зациклення» відбувається при ДПФ, оскільки алгоритм ДПФ вважає, що функція періодична. Існує безліч вагових вікон, названих на честь їх творців. Всі вони мають схожу форму і в значною мірою усувають розглянуті спотворення спектра. Наведемо формули двох вікон: Хеммінга і Блекмана (рис. 1.7):

$$w_{Hamming}[n] = 0{,}54 - 0{,}46\cos\frac{2\pi n}{N};$$

$$w_{Blackman}[n] = 0{,}42 - 0{,}5\cos\frac{2\pi n}{N} + 0{,}08\cos\frac{4\pi n}{N}.$$

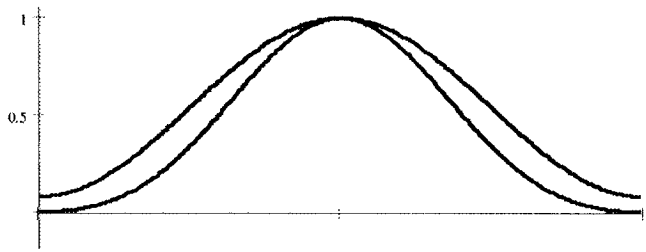


Рисунок 1.7 – Вагові вікна Хеммінга (верхнє) і Блекмана (нижнє)

В даному випадку вікно застосовується до сигналу з індексами від 0 до *N*. Вікно Хеммінга використовується найбільш часто. Вікно Блекмана має більш сильну дію при усуненні розглянутих спотворень, проте має й свої недоліки.

Важлива властивість спектрального аналізу полягає в тому, що не існує жодного, єдино правильного спектра будь-якого сигналу. Спектр можна обчислювати із застосуванням різних розмірів ШПФ і різних вагових вікон. Для кожної конкретної задачі необхідно використовувати свої способи.

Ще одна важлива властивість полягає в тому, що при розкладанні в спектр ми знаходимо не ті синусоїдальні складові, з яких складався вихідний сигнал, а лише знаходимо, з якими амплітудами потрібно взяти певні кратні частоти, щоб отримати вихідний сигнал. Проте зазвичай (особливо при використанні вагових вікон) цього майже не помітно за графіком спектра, тобто графік спектра досить адекватно відображає саме частоти вихідного сигналу.

1.5.4. Ортогональні перетворення в діадних базисах

Ортогональні перетворення (orthogonal transformation) в діадних (або двозначних знакозмінних) базисах визначені для даних, поданих векторами довжиною $N = 2^M$. До таких перетворень належать перетворення Адамара, Пелі, Уолша, Трахтмана і ряду інших. Матриця ядра кожного з подібних перетворень містить цілочислові коефіцієнти з множини {-1; +1}. Очевидно, що при виконанні подібних перетворень істотно скорочується обсяг обчислень за рахунок уникнення множення в кожній базовій операції.

Матриця ядра перетворення Уолша-Адамара для $N = 2^M$ може бути описана як результат кронекерівського добутку m матриць ДЕФ E_2 розміром 2×2:

$$A_N = A_{2^m} = E_2 \otimes E_2 \otimes E_2 \otimes \ldots \otimes E_2 = [E_2]^m = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^m,$$

де символ $\otimes$ – операція кронекерівського множення векторів, у результаті чого породжується матриця блокової структури. Помітимо, що операція кронекерівського множення двох матриць полягає в одержанні блокової матриці, блоками якої є помножена на відповідний елемент правої матриці ліва матриця, тобто:

$$A_4 = E_2 \otimes E_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \left|\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ - & - & - & - \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right|.$$

Матриця Адамара для $N = 8$

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Матриця ядра Адамара має такі властивості:

1) циклічність $a_{N+k} = a_k$; $a_{N-k} = a_{-k}$;

2) мультиплікативність $a_{k+1} = a_k * a_l$;

3) симетричність $A_N = A_N^T$.

У завданнях ЦОС використовуються також інші, подібні до перетворення Адамара, перетворення – Пелі, Уолша, Трахтмана та інших. Ядра (матриці) цих перетворень можуть бути отримані на основі матриці ядра перетворення Адамара при певному перевпорядкуванні рядків.

1.6 Згортка. Кореляція

Кореляція (correlation), і її окремий випадок для центрованих сигналів – коваріація (covariance), є методом аналізу сигналів. Цей математичний апарат знайшов застосування в обробці зображень у сфері комп'ютерного зору або дистанційного зондування з супутників, у яких порівнюються дані з різних зображень, у радарних або гідроакустичних установках для дальнометрії та місцевизначення, у яких порівнюються передані та відбиті сигнали.

Кореляція дозволяє визначити ступінь незалежності одного процесу від іншого або встановити подібність одного набору даних іншому. Кореляція також є невід'ємною частиною процесу згортання, що, за суттю, та ж кореляція двох послідовностей даних, при обчисленні якої одна з послідовностей згорнута в часі. Це означає, що для обчислення кореляції та згортки можуть використовуватися ті ж самі алгоритми.

Для двох послідовностей $\{x_k\}$ і $\{y_k\}$ довжини N з нульовим середнім значенням оцінювання їх взаємної кореляції здійснюється за формулою

$$\rho_{xy}(n) = \frac{r_{xy}(n)}{\sqrt{r_{xx}(0) \cdot r_{yy}(0)}}, \quad n = 0, \pm 1, \pm 2, \ldots$$

де $r_{xx}(0) = \frac{1}{N} \sum_{k=0}^{N-1} x_k^2$, $r_{yy}(0) = \frac{1}{N} \sum_{k=0}^{N-1} y_k^2$, $r_{xy}(n)$ – оцінка взаємної коваріації, що знаходиться за формулою

$$r_{xy}(n) = \frac{1}{N} \sum_{k=0}^{N-1} x_k y_{k+n} \quad .$$

Для послідовності $\{x_k\}$кінцевої довжини N з нульовим середнім значенням обчислення автокореляційної функції здійснюється таким чином

$$\rho_{xx}(n)=\frac{r_{xx}(n)}{r_{xx}(0)}, \qquad n=0,\pm1,\pm2,\ldots \ ,$$

де $r_{xx}(n)=\frac{1}{N}\sum_{k=0}^{N-n-1}x_k y_{k+n}$.

Існує кілька способів розрахунку відгуку системи на довільний вхідний сигнал. Найбільш розповсюджений спосіб розрахунку полягає в тому, що ми обчислюємо значення кожної точки в підсумковому сигналі як зважену суму певної множини сусідніх точок вихідного сигналу. Коефіцієнти цієї суми збігаються з імпульсною характеристикою лінійної системи, розгорнутої відносно точки 0. Звідси й береться формула згортки для одновимірного випадку:

$$y[n]=\sum_{k=-\infty}^{\infty}x[n-k]\cdot h[k] \ .$$

Розглянута операція отримання підсумкового сигналу за вхідним називається *згорткою* (compression). Отже, будь-яка лінійна система здійснює згортку вхідного сигналу зі своєю імпульсною характеристикою. Це записується так: $y[n]=x[n]*h[n]$. Функція $h[n]$ називається ядром згортки або імпульсною характеристикою лінійної системи.

Зазвичай всі сигнали, що обробляються на ЕОМ, мають кінцеву тривалість (тобто відмінні від нуля лише на кінцевому відрізку). Розглянемо, що відбувається з сигналом кінцевої тривалості, коли його згортають з кінцевим ядром згортки. Нехай сигнал $x[n]$ відмінний від нуля тільки на відрізку від 0 до N-1 включно («має довжину К»). Нехай ядро згортки $h[n]$ відмінне від нуля на відрізку від $-m_1$ до m_2 включно, що складається з M точок ($M=m_1+m_2+1$). Тоді при підстановці цих сигналів в рівняння згортки, отримаємо сигнал $y[n]$, який відрізняється від нуля на відрізку від $-m_1$ до $N-1+m_2$ включно. Таким чином, довжина підсумкового сигналу дорівнює $N+M-1$, тобто сумі довжин вихідного сигналу і ядра згортки мінус один. Отже, операція згортки розширює сигнал на M-1 точку, де M – довжина ядра згортки.

Властивості згортки:

1. Закон комутативності:

$x[n]*y[n]=y[n]*x[n]$ (тобто можна переставляти місцями вихідний сигнал і ядро згортки);

2. Закон асоціативності:

$(x[n]*y[n])*z[n]=x[n]*(y[n]*z[n])$ (тобто замість того, щоб проводити згортку по черзі в різних системах, можна отримати систему з ядром ($y[n]*z[n]$), яка є суперпозицією систем $y[n]$ і $z[n]$).

3. Закон дистрибутивності:

$x[n]*y[n]+x[n]*z[n]=x[n](y[n]+x[n])$.

Теорема згортки. Згортка в часовій області еквівалентна множенню в частотній області; множення в часовій області еквівалентно згортці в частотній області. Це означає, що для виконання згортки двох сигналів можна перевести їх в частотну область, помножити їх спектри і перевести їх назад в часову область. Така операція виглядає громіздко. Однак з появою алгоритмів ШПФ, що дозволяють швидко обчислювати перетворення Фур'є, обчислення згортки через частотну область стало широко використовуватися. При значних довжинах ядра згортки такий підхід дозволяє в сотні разів скоротити час обчислення згортки.

Циклічна згортка й кореляція

У дискретному вигляді лінійні перетворення можуть бути описані в загальному вигляді як векторно-матричні операції

$$Y = B_N X,$$

де X – вектор відліків вихідних даних, отриманий у результаті дискретизації безперервного сигналу відповідно до теореми Котельникова, Y – вектор відліків результату, B_N – матриця розміром $N \times N$, що визначає ядро перетворення.

До числа подібних перетворень належить циклічна згортка послідовностей $X=[x_0 \quad x_1 \quad x_2 \quad \dots \quad x_{N-1}]$ і $G=[g_0 \quad g_1 \quad g_2 \quad \dots \quad g_{N-1}]$ в цьому випадку будується матриця ядра згортки:

$$B_N^C = G_N = \begin{vmatrix} g_0 & g_{N-1} & g_{N-2} & \cdots & g_1 \\ g_1 & g_0 & g_{N-1} & \cdots & g_2 \\ g_2 & g_1 & g_0 & \cdots & g_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{N-1} & g_{N-2} & g_{N-3} & \cdots & g_0 \end{vmatrix}.$$

Кожний елемент вектора Y може бути описаний як

$$y_m = \sum_{n=0}^{N-1} g_{m-n} x_n .$$

Матриця ядра циклічної взаємокореляції може бути побудована як транспонована матриця ядра згортки, тобто таким чином:

$$K_N = B_N^K = \begin{vmatrix} g_0 & g_1 & g_2 & \cdots & g_{N-1} \\ g_{N-1} & g_0 & g_{N-1} & \cdots & g_{N-2} \\ g_{N-2} & g_{N-1} & g_0 & \cdots & g_{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_1 & g_2 & g_3 & \cdots & g_0 \end{vmatrix},$$

тому кожний відлік результату може бути записаний як

$$z_m = \sum_{n=0}^{N-1} g_{m+n} x_n;$$

причому $g_{m+n} = g_n$ для $m+n \geq N$.

Аперіодична згортка і кореляція на відміну від циклічної належить до класу локальних перетворень. При цьому, як правило, вважається, що розмір вектора вихідних даних значно більше розміру ядра згортки, що приводить до такого виразу для обчислення будь-якого відліку результату:

$$y_i = \sum_m g_m x_i - m \ .$$

Обчислення згортки і кореляції лежить в основі кореляційного методу заглушення завад. Сутність такого методу полягає у використанні розходження між кореляційними функціями сигналу та завади. Даний метод ефективний лише у випадку обробки періодичних або квазіперіодичних сигналів.

1.7 Цифрова фільтрація сигналів

Терміном цифровий фільтр називають апаратну або програмну реалізацію математичного алгоритму, входом якого є цифровий сигнал, а виходом – інший цифровий сигнал, форма якого і/або амплітудна та фазова характеристики спеціальним чином модифіковані.

В аналогових системах під фільтром розуміють деякий лінійний пристрій зі спеціальною частотною характеристикою $\dot{K}(j\omega)$, який перетворює вхідний сигнал $s(t)$ у вихідний $y(t)$ (рис. 1.8), заглушуючи або, навпаки, підсилюючи при цьому певні частоти в спектрі вхідного

сигналу. Вихідний сигнал $y(t)$ знаходиться як згортка вхідного сигналу $s(t)$ та імпульсної характеристики фільтра $h(t)$:

$$y(t) = s(t) * h(t) = \int_{-\infty}^{\infty} s(\tau)h(t-\tau)d\tau = \int_{-\infty}^{\infty} h(\tau)s(t-\tau)d\tau \quad s(t)$$

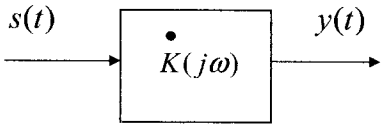


Рисунок 1.8 – Цифровий фільтр

За аналогією з аналоговим фільтром, цифровий фільтр (ЦФ, digital filter) перетворює послідовність відліків вхідного сигналу $\{s_k\}$ у числову послідовність вихідного сигналу $\{y_k\}$. Для ЦФ також вводять поняття імпульсної характеристики $\{h_k\}$, що є реакцією ЦФ на «одиничний імпульс (скачок)» тобто

$$(1,0,0,...) \xrightarrow{ЦФ} (h_0, h_1, h_2, ...) \quad .$$

Імпульсну характеристику (pulse response characteristic) $\{h_k\}$ ЦФ можна трактувати як результат дискретизації неперервної імпульсної характеристики $h(t)$ відповідного аналогового фільтра-прототипу (рис. 1.9).

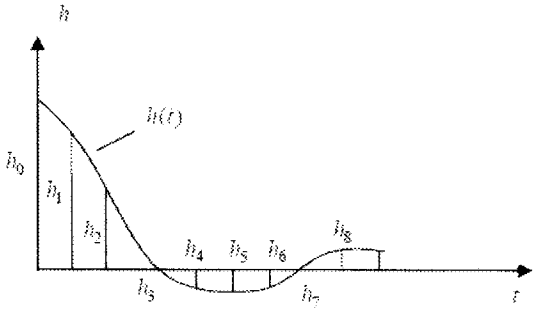


Рисунок 1.9 – Дискретизація імпульсної характеристики

Якщо взяти кінцеве число відліків $h(t)$, тоді отримаємо ЦФ із *кінцево-імпульсною характеристикою* (КІХ-фільтр, finite impulse response filtering). Якщо взяти нескінченне число відліків $h(t)$, отримаємо ЦФ із *нескінченно-імпульсною характеристикою* (НІХ-фільтр, infinite impulse response filtering).

Під фільтром зазвичай розуміють систему, що одні частоти пропускає, а інші затримує. Однак у техніці цифрової обробки сигналів поняття фільтра трактується більш широко. *Дискретним фільтром* називають довільну систему обробки дискретного сигналу, що має властивості лінійності й стаціонарності. Існують також фільтри зі змінними параметрами, наприклад, *адаптивні фільтри*, що змінюють свої параметри залежно від статистичних властивостей вхідного сигналу.

У загальному випадку, фільтр змінює в спектрі сигналу і амплітуди гармонік, і їх фази. Однак фільтри можна проектувати так, щоб вони не змінювали фазу сигналу. Такі фільтри називаються *фільтрами з лінійною фазою*. Це означає, що якщо вони і змінюють фазу сигналу, то роблять це так, що всі гармоніки сигналу зсуваються за часом на одну й ту ж величину. Таким чином, фільтри з лінійною фазою не спотворюють фазу сигналу, а лише зсувають весь сигнал в часі. Ядро згортки такого фільтра симетричне щодо своєї центральної точки.

Основна властивість будь-якого фільтра – це його частотна і фазова характеристики. Вони показують, як фільтр впливає на амплітуду і фазу різних гармонік оброблюваного сигналу. Якщо фільтр має лінійну фазу, то розглядається лише частотна характеристика фільтра. Зазвичай частотна характеристика зображається у вигляді графіка залежності амплітуди від частоти (в децибелах). Наприклад, якщо фільтр пропускає всі сигнали в смузі 0 ... 10 кГц без зміни, а всі сигнали в смузі вище 10 кГц заглушує в 2 рази (на 6 дБ), то частотна характеристика буде мати такий вигляд:

$$A(f) = \begin{cases} 0дБ, f < 10кГц \\ -6дБ, f > 10кГц \end{cases}.$$

Частотна характеристика в 0 дБ показує, що дані частоти фільтр пропускає без зміни. Ті частоти, амплітуда яких послаблюється фільтром в 2 рази, повинні мати амплітуду на 6 дБ менше. Тому їх амплітуда становить -6 дБ. Якщо фільтр посилює частоти, то його частотна характеристика на цих частотах є позитивною.

Вихідний сигнал *y(k)* фільтра, що має нетривіальну частотну характеристикою, залежить від декількох відліків вхідного сигналу *x(k)*. У загальному випадку при обчисленні вихідного відліку використовується також деяка кількість попередніх відліків вихідного сигналу. Для фільтрів, що не використовують вихідні відліки, рівняння фільтрації має вигляд

$$y(k) = \sum_{i=0}^{m} b_i x_{(k-i)} \quad .$$

Такі фільтри називаються *нерекурсивними* (nonrecursive filter) або трансверсальними. Кількість відліків m називається порядком фільтра. Структурна схема нерекурсивного фільтра показана на рис. 1.10. Імпульсна характеристика нерекурсивного фільтра визначається його коефіцієнтами $h(k) = b_k$. Оскільки в реальному пристрої кількість ліній затримки обмежена, а отже, і кількість коефіцієнтів, нерекурсивні фільтри відносять до класу КІХ-фільтрів.

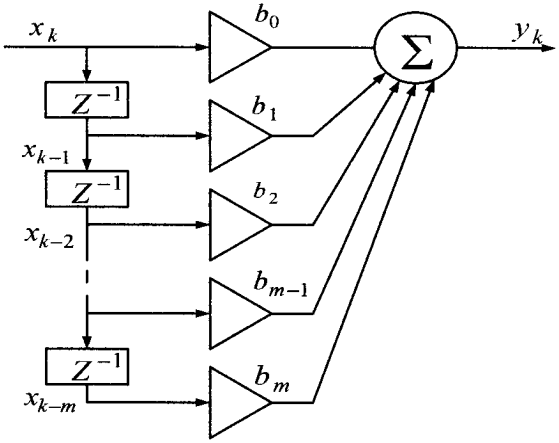


Рисунок 1.10 – Нерекурсивний фільтр

Фільтр, у якому також використовуються і вихідні відліки, називається *рекурсивним* (recursive filter) (рис. 1.11). Рівняння рекурсивного фільтра має вигляд

$$y(k) = \sum_{i=0}^{m} b_i x_{(k-i)} - \sum_{i=0}^{n} a_i y_{(k-i)} \quad .$$

Наявність у схемі рекурсивного фільтра зворотних зв'язків дозволяє одержати нескінченну імпульсну характеристику, тому такі фільтри належать до класу БІХ-фільтрів. Проте такі фільтри за деяких умов можуть бути нестійкими.

Під проектуванням (синтезом) цифрового фільтра розуміють вибір таких наборів його коефіцієнтів $\{a_i\}$ і $\{b_i\}$, які задовольняють задані вимоги. У завдання проектування входить також і вибір потрібної структури фільтра з урахуванням необхідної точності обчислень.

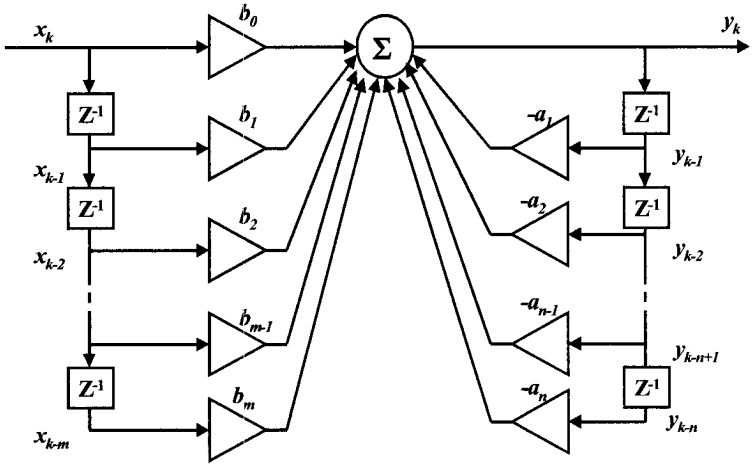


Рисунок 1.11 – Рекурсивний фільтр

У проектуванні та реалізації цифрових фільтрів застосовується безліч різноманітних підходів і методів, вибір яких залежить від багатьох факторів, зокрема – від того, як формулюється завдання фільтрації.

Найчастіше цифрова фільтрація застосовується для виділення сигналу або для відновлення сигналу. Виділення «корисного» сигналу необхідно, коли сигнал, що надходить у систему із зовнішнього середовища, змішаний із шумами, викликаними різноманітними фізичними процесами, що мають, як правило, випадковий характер. Відновлення сигналу необхідно через можливі спотворення сигналу, викликані роботою апаратури.

У зв'язку з тим, що теорія апроксимації ідеальних АЧХ аналоговими засобами добре розвинена, широке використання одержали методи синтезу цифрових фільтрів за аналоговими прототипами.

У прямих методах синтезу без використання аналогових прототипів часто використовується той факт, що ДПФ можна трактувати як обробку сигналу фільтром з відповідною імпульсною характеристикою.

Оскільки ДПФ лінійної згортки дорівнює добутку ДПФ послідовностей, що згортаються, алгоритм фільтрації в частотній області полягає в нижчевикладеному:

– Послідовність відліків вхідного сигналу та імпульсна характеристика фільтра доповнюються нулями так, щоб довжини послідовностей стали рівними і не меншими, ніж сума довжин вихідних послідовностей мінус одиниця.

– Обчислюються ДПФ доповнених нулями послідовностей.

– Обчислені ДПФ поелементно перемножуються.

– Обчислюється обернене ДПФ від результату перемножування.

Для підвищення ефективності фільтрація здійснюється в частотній області з використанням ШПФ.

При проектуванні цифрових фільтрів один із ключових моментів пов'язаний з особливостями технічної реалізації і рядом обмежень, обумовлених розрядністю цифрових обчислювальних пристроїв:

– шум квантування, що виникає при аналого-цифровому перетворенні;

– спотворення характеристик, що відбувається при квантуванні коефіцієнтів цифрових фільтрів;

– переповнення розрядної сітки в процесі обчислень;

– округлення проміжних результатів обчислень.

Тому при проектуванні цифрових фільтрів важливо правильно вибрати способи й формати подання чисел (дійсні або комплексні числа, з фіксованою або плаваючою крапкою тощо), динамічний діапазон подання даних, обумовлений розрядністю регістрів устаткування, а також оцінити можливий вплив шумів і спотворень.

Контрольні завдання та запитання

4. Які перетворення мають місце при цифровій обробці сигналів?
5. Що таке дискретний сигнал і дискретна послідовність?
6. У чому полягають взаємозв'язок і відмінність спектрів дискретного та аналогового сигналів?
7. Як за відомим спектром аналогового сигналу визначити спектр відповідного йому дискретного сигналу?
8. У чому полягає явище накладання спектрів при дискретизації сигналів?
9. Як здійснюється цифрове кодування сигналу?
10. Як визначається автокореляційна функція і спектральна щільність шуму квантування АЦП?
11. Відомо, що для отримання розбірливої людської мови її достаньо дискретизувати з частотою 8 кГц. Який діапазон частот може бути правильно переданий таким цифровим записом? Що необхідно зробити у разі дискретизації для правильної передачі частот цього діапазону?
12. Сигнал $x[n]$, відмінний від нуля на відрізку $[A,B]$, згортається з сигналом $h[n]$, відмінним від нуля на відрізку $[C,D]$. Знайти відрізок, на якому може бути відмінний від нуля підсумковий сигнал.
13. Розрахувати, скільки множень потрібно зробити для обчислення згортки сигналу довжини N з ядром довжини M.
14. Частота дискретизації сигналу дорівнює 44100 Гц. Розмір ШПФ дорівнює 4096. Яка довжина аналізованого блоку в секундах? За якими частотами (у герцах) буде розкладений сигнал?

15. Яку частотну роздільну здатність спектра ми отримаємо в попередньому прикладі? Який розмір ШПФ потрібно використовувати, щоб отримати частотну роздільну здатність близько 4 Гц?
16. Реалізувати знаходження і відображення спектра заданої ділянки сигналу. Ввести можливість вибору довжини сигналу, розміру ШПФ, виду вагового вікна.
17. Реалізувати швидку згортку двох сигналів через частотну область.
18. Реалізувати секційну згортку двох сигналів через частотну область.
19. Реалізувати алгоритм проектування фільтра за заданою частотною характеристикою. Спроектувати НЧ-фільтр з довільними параметрами.
20. Показати аналітично, що обернене ДПФ можна виконати за допомогою співвідношення (1.12).
21. Виконати чисельно ДПФ за формулами (1.7), (1.8), (1.12) і зрівняти відновлені сигнали. Розрахувати їх АЧХ і ФЧХ.
22. Обчислити обернене ДПФ відповідно до (1.13), використовуючи тільки інформацію про ФЧХ сигналу.
23. Чим відрізняється рекурсивний фільтр від нерекурсивного? Опишіть рівняння для обох фільтрів.
24. У чому полягає завдання проектування цифрових фільтрів? Які підходи при цьому використовуються?
25. Як виглядає алгоритм фільтрації із застосуванням ДПФ?
26. Які технічні обмеження впливають на характеристики проектованих фільтрів?

РОЗДІЛ 2 МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ

Методи обробки зображень (image processing) мають надзвичайно важливе значення в сучасній науці, вони є одними з таких, які постійно розвиваються та вдосконалюються. При цьому під обробкою зображень розуміють не лише поліпшення зорового сприйняття зображень, але й класифікацію об'єктів, що виконується при аналізі зображень.

Сфери застосування методів цифрової обробки в наш час значно розширюються, витісняючи аналогові методи обробки сигналів зображень. Методи цифрової обробки широко застосовуються в промисловості, мистецтві, медицині, космосі. Вони застосовуються при керуванні процесами, автоматизації виявлення об'єктів, розпізнаванні образів і в багатьох інших. Цифрова передача зображень із космічних апаратів, цифрові канали передачі сигналів зображень вимагають забезпечення передачі все більших потоків інформації. Формування зображень, поліпшення якості та автоматизація обробки медичних зображень, охоплюючи зображення, що створюються електронними мікроскопами, рентгенівськими апаратами, томографами тощо, є предметом сучасних досліджень та розробок. Автоматичний аналіз у системах дистанційного спостереження широко застосовується при аналізі місцевості, у лісовому господарстві, наприклад, для автоматичного підрахунку площі вирубок, у сільському господарстві для спостереження за дозріванням урожаю, у розвідці, у системах протипожежної безпеки. Контроль якості виробленої продукції виконується завдяки автоматичним методам аналізу сцен.

Сьогодні важко уявити сферу діяльності, у якій можна обійтися без комп'ютерної обробки зображень. При комп'ютерній обробці зображень вирішується широке коло таких завдань, як поліпшення якості зображень; вимірювання параметрів зображення; спектральний аналіз багатовимірних сигналів; розпізнавання зображень; стиск зображень.

2.1 Класичні методи обробки зображень

2.1.1 Математичні моделі зображень

Комп'ютерна обробка зображень можлива після перетворення сигналу зображення з неперервної форми в цифрову. Ефективність обробки залежить від адекватності моделі, що описує зображення, необхідної для розробки алгоритмів обробки. Модель зображення являє собою систему функцій, що описують істотні характеристики зображення: функцію яскравості, що відбиває зміну яскравості в площині зображення, просторові спектри й спектральні інтенсивності зображень, функції автокореляції. Канал зображення містить оптичну систему, оптико-

електричний перетворювач, пристрій аналого-цифрового перетворення (АЦП) і цифрової обробки сигналів зображення.

При обробці зображень широко використовується аналіз спектрів зображень. Спектр зображення (image spectrum) одержують прямим двовимірним перетворенням Фур'є функції, що описує зображення:

$$F(\omega_x, \omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp(-i(\omega_x x + \omega_y y))dxdy\,, \qquad (2.1)$$

де ω_x, ω_y – просторові частоти; $i = \sqrt{-1}$ – уявна одиниця.

Функція $\exp(-i(\omega_x x + \omega_y y))dxdy$ при фіксованих значеннях просторових частот описує плоску хвилю в площині зображення (x, y) (відповідно до рисунка 2.1).

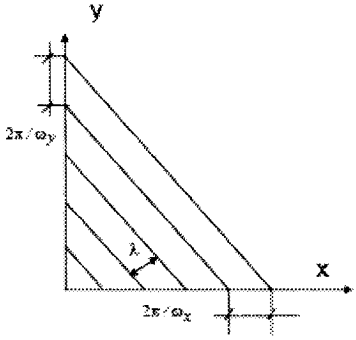


Рисунок 2.1 – Визначення просторових частот зображення

Формула (2.1) пов'язує дійсну функцію, що описує яскравість зображення f(x, y), з комплексною функцією частоти – спектром зображення $F(\omega_x, \omega_y)$:

$$F(\omega_x, \omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\cos(\omega_x x + \omega_y y)dxdy + \\ + i\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(-f(x,y))\sin(\omega_x x + \omega_y y)dxdy = \mathrm{Re}(\omega_x, \omega_y) + i\,\mathrm{Im}(\omega_x, \omega_y), \qquad (2.2)$$

де $\mathrm{Re}(\omega_x, \omega_y)$ – дійсна частина спектра; $\mathrm{Im}(\omega_x, \omega_y)$ – реальна частина спектра.

Амплітуда і фаза спектра визначаються за формулами (2.3) і (2.4) відповідно:

$$F(\omega_x,\omega_y)=\sqrt{\mathrm{Re}(\psi_x,\psi_y)^2+\mathrm{Im}(\psi_x,\psi_y)^2}\,, \tag{2.3}$$

$$\varphi(\psi_x,\psi_y)=arctg(\mathrm{Im}(\psi_x,\psi_y)/\mathrm{Re}(\psi_x,\psi_y))\quad .$$

Із (2.3)

$$F(\omega_x,\omega_y)=F(\psi_x,\psi_y)\exp(i\varphi(\omega_x,\omega_y))\quad . \tag{2.4}$$

Обернене перетворення Фур'є дозволяє відновити зображення за його спектром:

$$f(x,y)=(1/4\pi^2)\int_{-\infty}^{\infty}\int F(\psi_x,\psi_y)\exp(i\varphi(\omega_x,\omega_y))\quad . \tag{2.5}$$

Спектральна інтенсивність (spectral intensity) зображення характеризує розподіл енергії по просторових частотах. Вона визначається як квадрат модуля спектра зображення:

$$S(\omega_x,\omega_y)=\mathrm{Re}(\omega_x,\omega_y)^2+i\,\mathrm{Im}(\omega_x,\omega_y)^2=F^2(\omega_x,\omega_y)\quad . \tag{2.6}$$

Для її назви використовуються терміни «спектральна щільність» (spectral density) і «енергетичний спектр» (energy spectrum).

Енергія зображення визначається як інтеграл енергетичного спектра за просторовими частотами. Відповідно до теореми Парсеваля енергія зображення може бути обчислена відповідно до (2.7):

$$\int_{-\infty-\infty}^{\infty}\int^{\infty} f^2(x,y)dxdy=\int_{-\infty-\infty}^{\infty}\int^{\infty}\left|F(\omega_x,\omega_y)\right|^2 d\omega_x d\omega_y\quad . \tag{2.7}$$

Імовірнісні моделі зображень (probabilistic image models) широко використовуються для опису зображень. Зображення в цьому випадку розглядається як випадкова функція просторових координат (x,y) і часу t. Випадковий процес називається стаціонарним у широкому сенсі, якщо він має постійні значення математичного очікування й дисперсії, а його автокореляційна функція залежить не від координат, а від їх різниці (зсуву). Випадковий процес називається стаціонарним у вузькому сенсі, якщо його n-вимірна щільність розподілу ймовірностей інваріантна до зсуву. Випадковий процес описується щільністю розподілу ймовірності яскравості в зображенні по просторових координатах для деякого фіксованого моменту часу t $p(x,y)$.

У відповідності з визначенням математичне сподівання (середнє значення, mathematical expectation) стаціонарного процесу в широкому розумінні

$$Mf = \xi = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)p(x,y)dxdy = const \quad . \tag{2.8}$$

Дисперсія (dispersion, variance)

$$Df = \sigma^2 = E(f(x,y)-\xi)^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(f(x,y)-\xi)^2 p(x,y)dxdy = const \tag{2.9}$$

Функція автокореляції (autocorrelation function) обчислюється таким чином:

$$R(\tau_x,\tau_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)f(x-\tau_x, y-\tau_y)dxdy \quad , \tag{2.10}$$

де τ_x, τ_y задають зсуви зображення по відповідних осях координат.

2.1.2 Статистичні методи аналізу зображень

Перші статистичні методи були основані на аналізі взаємного розташування відтінків сірого кольору зображення та частоти їх появи, що характеризувалась двовимірною функцією щільності ймовірності. Для класифікації та розпізнавання текстур за допомогою функції щільності ймовірності було визначено чотири характерних ознаки:

- кутовий момент;
- контрастність;
- кореляція;
- ентропія.

До статистичних методів можна віднести кореляційний аналіз зображень. На основі кореляційного підходу найбільш успішним є метод матриць взаємозв'язку (co-occurrence matrices – GLCM). Дані матриці характеризують частоту пар різних градацій сірого кольору, що присутні в зображенні, і визначаються шляхом кореляційного аналізу пікселів зображення, при цьому якщо піксель відповідає вибраній градації, то він враховується як одиничне значення, якщо ні, то як нульове. У випадку кольорових зображень даний підхід використовують до аналізу кожного з трьох базових кольорів. Як слідує з методу формування матриць взаємозв'язку, вони найкраще підходять до розв'язання задач класифікації текстур.

Наряду зі статистичним аналізом в просторовій області використовують аналогічний аналіз в спектральній області із застосуванням двовимірного дискретного перетворення Фур'є в базисі різних ортогональних функцій – дискретних експоненціальних функцій (ДЕФ), косинусних, Уолша-Адамара, Хаара та інших. Для аналізу текстур використовують метод гістограм розподілу спектральних коефіцієнтів. Загальною характеристикою такого підходу є те, що гістограми спектральних коефіцієнтів, що отримані в результаті інтегральних перетворень, більш стійкі та надійні, ніж гістограми розподілу окремих пікселів чи груп пікселів. Даний метод також не чутливий до присутнього в зображенні шуму, але це є одночасно і недоліком в разі застосування методу до розв'язання задачі розпізнавання незначних за розміром об'єктів на текстурованому фоні. Також спектральні методи згладжують різкі границі між об'єктами зображення.

Статистичний аналіз пікселів та спектральних коефіцієнтів на основі гістограм в основному використовують до класифікації статичних текстур, до аналізу динамічних текстур використовують методи на основі кореляційного аналізу.

Найпростішою і найбільш часто вживаною є стохастична модель. Фон зображення характеризують гістограмою розподілу значень кольору за величиною в деякій базовій області, вільній від об'єктів. Гістограму апроксимують функцією щільності розподілу ймовірності, найбільш часто гаусовою. В цьому випадку параметрами моделі є середнє значення m та дисперсія σ. За максимальне відхилення сигналу моделі приймають 2σ. За мінімальне порогове значення величини відхилення можна прийняти величину $\varepsilon_{tr} = 3\sigma$.

Як приклад розглянемо алгоритм розпізнавання зображення об'єктів за допомогою стохастичної моделі:

$$\textbf{if } |u_{i,j} - m| > \varepsilon_{tr} \textbf{ then } v_{i,j} = u_{i,j} \textbf{ else } v_{i,j} = 0 \ , \qquad (2.11)$$

де $u_{i,j}$, $v_{i,j}$ – елементи матриць заданого зображення та зображення об'єктів.

Метод побудови моделі та її застосування викладено нижче.

1. Фрагмент зображення фону, вільний від сторонніх предметів, розміром $n_x \times n_y = 80 \times 80$ пікселів використано як базовий для того, щоб знайти статистичні параметри фону.

2. За допомогою виразу (2.11) створено зображення об'єктів.

На рис. 2.2 наведено результат розпізнавання та відбору об'єктів згідно з алгоритмом (2.11) за статистичним методом. Як видно з рисунка, всі сторонні об'єкти, як великі (листки), так і малі (плями білого кольору)

виділені, але присутній шум похибки. Зменшити похибку можна збільшивши порогове значення відбору, але, як показали чисельні експерименти, в цьому випадку втрачається якість зображення об'єктів. Таким чином, стохастична модель погано апроксимує фоновий сигнал та неспроможна повністю відділити шукані об'єкти від фону.

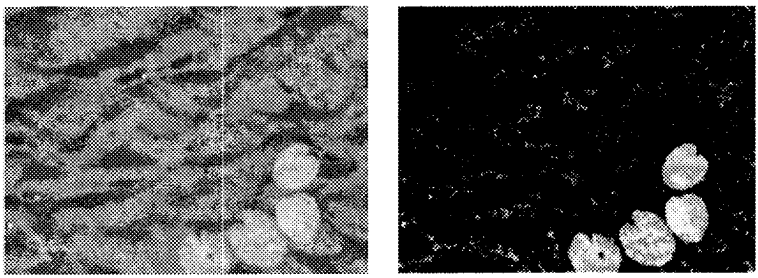


Рисунок 2.2 – Тестове зображення та результат розпізнавання об'єктів за статичним методом

2.1.3 Фільтрація зображень

Зазвичай зображення, сформовані різними інформаційними системами, спотворюються дією завад. Це ускладнює як їхній візуальний аналіз, так і автоматичну обробку. При вирішенні деяких завдань обробки зображень у ролі завад можуть виступати ті або інші компоненти самого зображення. Наприклад, при аналізі космічного знімка земної поверхні може стояти завдання визначення границь між її окремими ділянками – лісом і полем, водою й сушею тощо. З погляду цього завдання окремі деталі зображення всередині розділених ділянок є завадою.

Ослаблення дії завад досягається фільтрацією. При фільтрації яскравість (сигнал) кожної точки вихідного зображення, спотвореного завадою, замінюється деяким іншим значенням яскравості, яке меншою мірою було спотворене завадою. Фільтрація зображень здійснюється в просторовій і частотній областях. При просторовій фільтрації зображень перетворення виконується безпосередньо над значеннями відліків зображення. Результатом фільтрації є оцінка корисного сигналу зображення. Це досягається завдяки тому, що зображення часто являє собою двовимірну функцію просторових координат, що змінюється по цих координатах повільніше, ніж завада, що також є двовимірною функцією. Це дозволяє при оцінюванні корисного сигналу в кожній точці зображення взяти до уваги сусідні точки, скориставшись певною подібністю сигналу. В інших випадках, навпаки, ознакою корисного сигналу є різкі перепади яскравості. Однак, як правило, частота цих перепадів відносно невелика, так що на значних проміжках сигнал або постійний, або змінюється

повільно. І в цьому випадку властивості сигналу проявляються при спостереженні не тільки його окремої точки, але й при аналізі її околу. Поняття околу є досить умовним. На рисунку 2.3 наведена ієрархія околів відліку, позначеного «0».

«1» позначений окіл першого порядку, для якого відстань між елементами дорівнює 1. «2» позначений окіл другого порядку, до якого відносять діагональні елементи, відстань від яких до центрального відліку «0» дорівнює $\sqrt{2}$. Окіл третього порядку представлений елементами, що знаходяться від центрального елемента на відстані 2, і так далі.

9	8	7	6	7	8	9
8	5	4	3	4	5	8
7	4	2	1	2	4	7
6	3	1	0	1	3	6
7	4	2	1	2	4	7
8	5	4	3	4	5	8
9	8	7	6	7	8	9

Рисунок 2.3 – Конфігурації околу елемента «0» у кадрі зображення в ієрархічній послідовності

Відповідно до рисунка 2.3 формується ієрархія конфігурацій околу центрального відліку розглянутого фрагмента кадру за зростанням відстаней від нього до відліку околу. Окіл може бути утворений лише найближчими сусідами, але може містити й досить багато елементів кадру. При розгляді околу великого розміру іноді встановлюється різний ступінь впливу далеких і близьких від центра околу точок на сигнал, формований на виході фільтра в даній точці кадру. Таким чином, ідеологія фільтрації ґрунтується на використанні як даних поточної точки, так і її околу.

Традиційна фільтрація в частотній області вимагає виконання такої послідовності перетворень:

- двовимірне дискретне перетворення зображення із просторової області в частотну (наприклад, за допомогою дискретного перетворення Фур'є),
- перетворення дискретного спектра сигналу зображення,
- обернене двовимірне дискретне перетворення, що дозволяє відновити корисний сигнал зображення в просторовій області.

Завдання полягає в тому, щоб знайти таку обчислювальну процедуру, що забезпечила б одержання найкращих результатів.

Використання фільтрів для аналізу, текстурованих зображень аналогічно спектральному аналізу, але має ряд переваг в тих випадках, коли характерні ознаки спектра зображення відомі. Фільтрація дозволяє більш точно розділити сигнал на складові, що відповідають різним частотним смугам.

2.1.3.1 Оптимальна лінійна фільтрація

Нехай $x_{i,j}$ – значення яскравості зображення – корисного сигналу на перетині i-го рядка та j-го стовпця, а зображення, що знаходиться на вході фільтра, описується моделлю:

$$y_{i,j} = f(x_{i,j}, n_{i,j}), \quad i = \overline{0, I-1}, \quad j = \overline{0, J-1}, \tag{2.12}$$

де $n_{i,j}$ – значення завади в точці з координатами (i, j), $f(\cdot)$ – функція, що описує взаємодію сигналу і завади, а I та J – відповідно число рядків і стовпців у кадрі.

Надалі будемо дотримуватися прийнятої при цифровій обробці зображень декартової системи координат з початком у лівому верхньому кутку кадру та з додатними напрямками із цієї точки вниз та вправо.

При лінійній фільтрації вихідний ефект визначається лінійною комбінацією вхідних даних:

$$x(i, j) = \sum_{(i_1, j_1) \in S} \sum a(i_1, j_1) \cdot y(i - i_1, j - j_1). \tag{2.13}$$

В цьому рівнянні $x(i, j) = x_{i,j}$ – результат фільтрації корисного сигналу в точці кадру з координатами (i, j); S – множина координат точок, що утворюють окіл; $a(i_1, j_1)$ – вагові коефіцієнти, сукупність яких являє собою двовимірну *імпульсну характеристику* (ІХ). Якщо область кінцева, то імпульсна характеристика має кінцеву довжину й фільтр називається КІХ-фільтром. В іншому випадку імпульсна характеристика має нескінченну довжину, а фільтр назву НІХ-фільтра. У виразі (2.13) прийнято, що ІХ не залежить від координат точки, у якій визначається вихідний ефект. Процедури обробки зображень, що мають властивість незалежності від координат, називаються *однорідними.*

Найбільш поширеним критерієм оптимальності, який використовується для оцінювання якості обробки зображень, є критерій мінімуму середньоквадратичної похибки. Для фільтрації запишемо його так:

$$E\left\{\left[x(i,j)-\sum_{(i_1,j_1)\in S}\sum a(i_1,j_1)\cdot y(i-i_1,j-j_1)\right]^2\right\}=\min_{a(\cdot)}, \qquad (2.14)$$

де $E\{\}$ – символ математичного сподівання. Згідно з (2.14) пошук оптимального фільтра полягає у визначенні його ІХ таким чином, щоб середній квадрат похибки $\varepsilon(i,j)=x(i,j)-x^*(i,j)$, який виражає різницю між сигналом $x(i,j)$ і його оцінкою $x^*(i,j)$, яка формується фільтром, був мінімальним. Математичне сподівання обчислюється за всіма випадковими величинами, які є в (2.14), що свідчить про орієнтацію критерію на врахування середніх похибок.

Розглянемо реалізацію лінійної фільтрації в середовищі Matlab.

Синтаксис:

```
D=filter2(h,S)
D=filter2(h,S, shape)
```

Вбудована функція D=filter2(h,S) виконує фільтрацію даних, заданих у двовимірному масиві S, двовимірним КІХ-фільтром, коефіцієнти якого зведені в матрицю h, яку також називають маскою фільтра. Як правило, D і S є напівтоновими зображеннями. Результат фільтрації, що повертається в матриці D, обчислюється як двовимірна згортка, яка виконується за допомогою функції conv2.

Результат роботи функції D=filter2(h,S, shape) залежить від значення параметра shape. Якщо параметру присвоєно значення «same», то в D повертається центральна частина згортки, розмір якої дорівнює розміру вихідних даних – S. Цей параметр використовується за замовчуванням. Якщо параметру присвоєно значення «full», то повертається весь результат згортки. При цьому size(D)>size(S). Якщо параметру присвоєно значення «valid», повертається лише та частина згортки, при обчисленні якої не використовувались крайові частини вихідних даних, доповнені нулями. При цьому size(D)<size(S).

В залежності від складності зображення використовують два або більше двовимірних фільтри з коротко-імпульсною характеристикою (КІХ) або банки вузькосмугових фільтрів виду

$$s_{i,j}=-\sum_{m,n=0}^{N-1,M-1}h_{m,n}u_{i-m,j-n}, \qquad (2.15)$$

де $s_{i,j}$ та $u_{i-m,j-n}$ – відповідно, фільтрований сигнал та сигнал зображення, $h_{m,n}$ – імпульсна характеристика розміром $M\times N$. Як

критерій оптимальності використовують співвідношення середніх значень сигналів на виході фільтра, що отримані для текстури, яка відповідає характеристиці фільтра, та для текстур, які не відповідають характеристиці фільтра.

Для фільтрації зображень на основі їх спектральних ознак використовують фільтри Габора. Імпульсна характеристика фільтра має вигляд

$$h_{i,j} = \frac{1}{2\pi\sigma_x\sigma_y} exp\left(\frac{i^2}{2\sigma_x} + \frac{j^2}{2\sigma_y}\right) exp\left(2\pi\left(f_x i + f_y j\right)\right). \qquad (2.16)$$

Вона поєднує гаусову функцію та гармоніку, що характеризує частотну характеристику зображення. За допомогою параметрів дисперсії σ^2 в (2.16) регулюють смугу частотного фільтра з парою центральних частот (f_x, f_y) за двома просторовими координатами. Як правило, для характеристики текстурованого зображення використовують набір фільтрів з фіксованими або визначеними для кожної текстури парами частот.

Алгоритм класифікації та розпізнавання зображень охоплює такі етапи:

1) фільтрація зображення за допомогою набору фільтрів;

2) обчислення потужності на виході кожного з фільтрів;

3) згладжування значень потужності;

4) нормалізація значень потужності;

5) класифікація (розпізнавання) за допомогою шаблонів або співвідношень значень потужності.

Недоліком даного методу є те, що він потребує значного обсягу обчислень в тому разі, коли розмір фільтра $M \times N$ значний і число фільтрів велике.

У практиці цифрової обробки зображень широко використовується маскова фільтрація. Її лінійний різновид є одним з варіантів двовимірної фільтрації з кінцевою імпульсною характеристикою (КІХ) фільтра. Як маска використовується безліч вагових коефіцієнтів, заданих у всіх точках околу S , що зазвичай симетрично оточують поточну точку кадру.

2.1.3.2 Нелінійна фільтрація

У результаті застосування лінійних згладжувальних фільтрів відбувається заглушення шумів, але одночасно розмиваються границі між областями з різною амплітудою сигналу. Для зменшення кількості розмитих границь розроблені різні нелінійні фільтри. Як і лінійні КІХ-фільтри, нелінійні фільтри працюють у ковзному вікні. Різниця між

лінійною та нелінійною фільтрацією полягає в тому, що при лінійній фільтрації обчислюється лінійна комбінація відліків сигналу, а при нелінійній фільтрації виконуються нелінійні перетворення відліків сигналу в околі елементів, які зумовлюються маскою фільтра.

Сігма-фільтр

Сігма-фільтр призначений для заглушення шумів у зображенні зі збереженням контурів (різких границь областей). Центральний елемент маски заміщується зваженим середнім значенням, обчисленим лише за тими амплітудами відліків, значення яких потрапляють в $\pm k\sigma$ – область, яка вибрана згідно з яскравістю центрального елемента. σ вибирається або як середньоквадратичне відхилення (СКВ) заглушуваного шуму, або як СКВ в масці, або встановлюється рівною СКВ, отриманому на всьому зображенні:

$$g(x,y) = \sum_{s,t \in S} h(s,t) f(x-s, y-t), \qquad (2.16)$$

де S-окіл становлять ті значення координат маски, у яких виконується накладена умова:

$$S = \{(s,t) : |f(x-s, y-t) - f(x-y)| \le k\sigma\}, \qquad (2.17)$$

$h(s,t)$ – КІХ лінійного згладжувального фільтра.

При k = 2 діапазон значень, які замінюються, становить $\pm 2\sigma$, у випадку нормального розподілу шуму ймовірність потрапляння амплітуди за межі діапазону дорівнює 4,55%.

Медіанний фільтр

Медіанний фільтр (МФ) замінює центральний елемент маски медіаною впорядкованої вибірки, сформованої зі всіх амплітуд відліків, що покриваються маскою фільтра. При застосуванні МФ відбувається послідовна обробка кожної точки кадру, у результаті чого утворюється послідовність оцінок. При медіанній фільтрації використовується ковзне двовимірне вікно. У принципі, для кожного відліку виконується незалежне оцінювання медіани у вікні. З метою прискорення оцінювання доцільно алгоритмічно на кожному кроці використовувати раніше виконані обчислення. Розмір вікна встановлюється непарним і рівним $m \times n$. Відліки зображення, що знаходяться в межах вікна, утворюють робочу вибірку поточного відліку. Якщо впорядкувати послідовність $\{f_i, i = [1, mn]\}$ за зростанням, то її медіаною буде той елемент вибірки, що займає центральне місце в цій упорядкованій послідовності. Цей елемент є

$(mn+1)/2$ найбільшим і $(mn+1)/2$ найменшим значенням у вибірці й визначає результат медіанної фільтрації для поточної точки кадру. Введемо позначення описаної процедури у вигляді:

$$g_{med} = med(f_1, f_2, ..., f_n). \tag{2.18}$$

Розглянемо приклад. Припустимо, що впорядкована послідовність Y у вікні розміром 3×3 має вигляд: Y={76,100,69,120,210,143,87,130,155}, де елемент 210 відповідає центру вікна (x, y). Велике значення яскравості в цій точці кадру є результатом впливу імпульсної завади. Упорядкована за зростанням вибірка має вигляд: {69,76,87,100,120,130,143,155,210}, отже, відповідно до розглянутої вище процедури (2.18), на виході медіанного фільтра одержуємо $g_{med} = 120$. Бачимо, що врахування яскравостей елементів околу при фільтрації в поточній точці призвело до заглушення імпульсної завади. Якщо імпульсна завада не є точковою, а займає деяку область, то вона також може бути подавлена, якщо розмір цієї локальної області буде менше, ніж половина розміру апертури МФ. Тому для заглушення імпульсних завад, що вражають локальні ділянки зображення, варто збільшувати розміри апертури МФ.

З (2.8) слідує, що дія МФ полягає в «ігноруванні» як позитивних, так і негативних викидів значень вхідної вибірки. Такий принцип заглушення завад може бути застосований і для ослаблення шуму на зображенні. Однак дослідження заглушення шуму за допомогою медіанної фільтрації показує, що її ефективність при вирішенні цього завдання нижча, ніж лінійної фільтрації. Медіанна фільтрація краще зберігає границі зображення, ніж будь-яка лінійна фільтрація.

Медіанні фільтри заглушають імпульсні шуми. До таких шумів належить шум типу «сіль і перець», відліки якого мають значення, що відповідають максимальному («сіль») і мінімальному («перець») рівням квантування в сигналі зображення. Різкі зміни амплітуди зберігаються медіанним фільтром, а імпульсна завада, розмір якої $\leq mn/2$, таким фільтром заглушається. Однак при збільшенні маски фільтра можна втратити інформацію про малорозмірні області зображення та отримати спотворення границь областей, особливо в кутових сегментах зображення.

Оскільки застосування МФ призводить до заглушення високих частот зображення, викликаючи розмивання країв і текстур, все більший розвиток отримують схеми адаптивної фільтрації, які дозволяють змінити імпульсну характеристику фільтра залежно від локального значення сигналу зображення.

Один з алгоритмів **адаптивної медіанної фільтрації (АМФ)** виконується таким способом. У вікні фільтрації оцінюються мінімальне значення сигналу $f_{\min}$, максимальне значення $f_{\max}$ і медіана f_{med}. Фільтрації

піддається тільки той центральний елемент вікна f (x, y), для якого виконується умова 2.19 а): значення медіани більше мінімального й менше максимального значень у вікні й не виконується умова 2.19 б): значення сигналу в центрі вікна більше мінімального й менше максимального значень у вікні.

$$\text{а)}\ A1 = f_{med} - f_{\min}; A2 = f_{med} - f_{\max}; A1 > 0, A2 < 0,$$
$$\text{б)}\ B1 = f(x,y) - f_{\min}; B2 = f(x,y) - f_{\max}; B1 > 0, B2 < 0. \qquad (2.19)$$

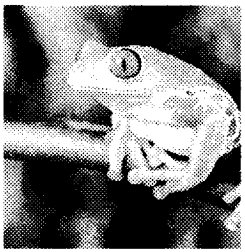
Застосування такого фільтра дозволяє видалити біполярну імпульсну заваду, забезпечити згладжування шумів і зменшити заглушення високих частот у зображенні.

На рисунку 2.4 наведено приклад усунення шуму за допомогою медіанного фільтра.

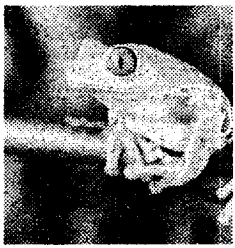
У системі Matlab (Image Processing Toolbox) існує можливість формування й накладання на зображення трьох типів шумів. Для цього використовується вбудована функція imnoise, що призначена, в основному, для створення тестових зображень, які використовуються при виборі й дослідженні методів фільтрації шуму. В даному випадку на зображення був накладений імпульсний шум за допомогою команди

```
J = imnoise(I,'salt & pepper',0.02);
```

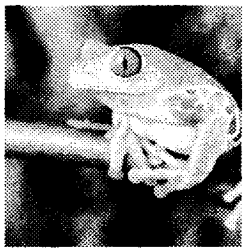
Для наочного порівняння наведемо три зображення разом: вихідне, зашумлене та відновлене. Як видно з рисунка 2.4, вихідне та відновлене зображення майже не відрізняються один від одного.



Вихідне зображення



Зашумлене зображення (імпульсний шум)



Відновлене зображення

Рисунок 2.4 – Приклад застосування медіанної фільтрації

2.1.3.3 Інверсні фільтри в задачах обробки зображень

Інверсна фільтрація (inverse filtering) широко застосовується в обробці зображень для розв'язання таких задач, як відновлення зображень, що вражені шумом, усунення розмиття, покращення контрастності. В задачах розпізнавання об'єктів та образів інверсні фільтри використовують як допоміжні засоби для покращення зображення в цілому перед операцією розпізнавання. Задача лінійної інверсної фільтрації полягає в тому, щоб за допомогою даних вимірювань у вигляді матриці елементів зображення $s_{i,j}$ відновити оригінальну матрицю зображення $u_{i,j}$ за умови, що матриці пов'язані деяким оператором з імпульсною характеристикою $h_{m,n}$ розміром $M \times N$ і присутній адитивний шум $\xi_{m,n}$ з відомою кореляційною характеристикою. Формально задачу можна записати у вигляді рівняння

$$s_{i,j} = \sum_{m,n=0}^{N-1,M-1} h_{m,n} u_{i-m,j-n} + \xi_{i,j}. \qquad (2.20)$$

Метод розв'язання рівняння (2.20), що став класичним, дав Н. Вінер, його представлено функцією Matlab *wiener2*. Метод оснований на перетворенні Фур'є. Однак застосування методу має ряд недоліків:

- в більшості прикладних задач неможливо виділити окремо шум та визначити його спектрально-кореляційну характеристику;
- визначення спектральної характеристики оригінального сигналу виконується за допомогою операції ділення, що робить метод чутливим до параметрів шуму та точності визначення імпульсної характеристики;
- для розв'язання рівняння (2.20) потрібно знайти оцінку імпульсної характеристики, що є самостійною непростою задачею.

У зв'язку з означеними проблемами часто використовують більш простий підхід до розв'язання рівняння (2.20), що нехтує складовою шуму. Для знаходження матриці оригінального зображення U використовують метод дискретного перетворення Фур'є (ДПФ). Позначимо оператор ДПФ як $\Phi(\cdot)$, обернений $\Phi^{-1}(\cdot)$, матрицю даних зображення як S. Тоді

$$U = \Phi^{-1}\left(\left(f + W\Phi(S)\right)^{-1}\right),$$

де W – функція спектрального вікна, $|f| << 1$ – функція регуляризації.

Слід відзначити, що розмір імпульсної характеристики, як правило, рівний розміру зображень, тому перетворення ДПФ виконують з використанням всієї множини елементів матриць зображень. Це є причиною того, що метод інверсної фільтрації не застосовують до розв'язання задач розпізнавання об'єктів та аналізу динамічних текстур як

основний засіб фільтрації ознак.

Розглянемо задачу інверсної фільтрації з точки зору ідентифікації динамічних систем і сформулюємо її у формі, що дозволить виконувати фільтрацію ознак динамічних текстур.

Нехай задана двовимірна функція Хевісайда

$$e_{ij} = \begin{cases} 1, \ (i \geq 0^+) \wedge (j \geq 0^+); \\ 0.5, \ i = 0, j = 0; \\ 0, \ (i \leq 0^-) \vee (j \leq 0^-), \end{cases} \tag{2.21}$$

зображення текстурованого фону задано в області $(i \geq 0^+) \wedge (j \geq 0^+)$ і є відгуком лінійної динамічної системи на збудження виду (2.21), його можна подати як

$$u_{i,j} = \sum_{m,n=0}^{P,Q} h_{m,n} e_{i-m,j-n} + \xi_{i,j}, \tag{2.22}$$

де $h_{m,n}$ – перехідна характеристики системи розміром $(P+1)\times(Q+1)$, що значно менший розміру зображення. Поставимо задачу відтворення за допомогою інверсного фільтра з перехідною характеристикою $\tilde{h}_{m,n}$ сигналу збудження в (2.22), що є постійною величиною в межах зображення. Тоді

$$\sum_{m,n=0}^{P,Q} \tilde{h}_{m,n} u_{i-m,j-n} = E + \varsigma_{i,j}, \tag{2.23}$$

де E – константа, $\varsigma_{i,j}$ – відліки шуму, що пов'язані з наближеним характером інверсного фільтра щодо сигналу фону.

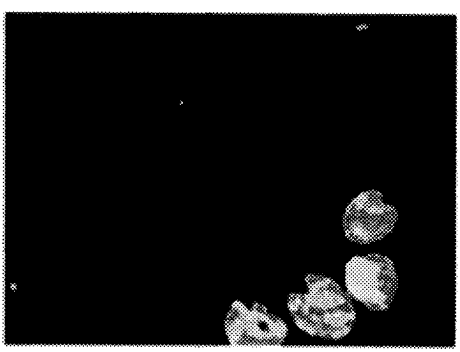


Рисунок 2.5 – Розпізнавання об'єктів за методом інверсної фільтрації

Розглянемо даний метод на прикладі, за тестове зображення використаємо рис. 2.2. Як видно з рисунка 2.5, зображення практично вільне від зайвих елементів, великі об'єкти не спотворені. В результаті фільтрації втрачено найменший світлий елемент, але виділено темний елемент біля верхньої границі зображення.

2.1.4 Методи на основі динамічних моделей

Для аналізу зображень використовують двовимірні дискретні лінійні та нелінійні моделі, що описують динаміку поверхні зображення у просторі. В найбільш загальній формі лінійну модель описують системою рівнянь виду

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{v}(t), \quad \mathbf{v}(t) \approx N(0,Q); \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{w}(t), \quad \mathbf{w}(t) \approx N(0,R) \end{cases}, \tag{2.24}$$

де $\mathbf{y}(t)$ – вектор зображення, $\mathbf{x}(t)$ – вектор стану динамічної моделі, $\mathbf{A}$ – матриця-оператор моделі має таку структуру

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -a_M & -a_{M-1} & \cdots & -a_1 \end{bmatrix}, \tag{2.25}$$

M – порядок моделі, a_i – параметри оператора моделі, $\mathbf{B}$ – матриця фільтра ковзного середнього, $\mathbf{C}$ – матриця функціонального перетворення при вимірюванні значень елементів зображення, $\mathbf{v}(t)$, $\mathbf{w}(t)$ – вектори гаусового центрованого шуму з кореляційними функціями Q та R, $\mathbf{x}_0$ – вектор початкових або граничних значень.

Систему рівнянь широко використовують в системах керування. Методи на основі рівняння стану (2.24) допускають, що можуть існувати численні динамічні текстури в різних областях зображення. Їх визначають як локальні, що відповідають невеликим просторовим регіонам. Локальні текстури мають різні динамічні властивості, що задані відповідними рівняннями стану. На рис. 2.6 наведено приклад зображення, що поділено на блоки. Кожний окремий блок має власну динамічну модель. На рис. 2.7 показано три типи імпульсних характеристик локальних моделей.

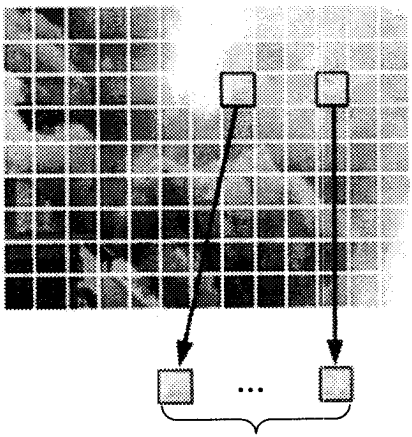


Рисунок 2.6 – Приклад локального динамічного фону

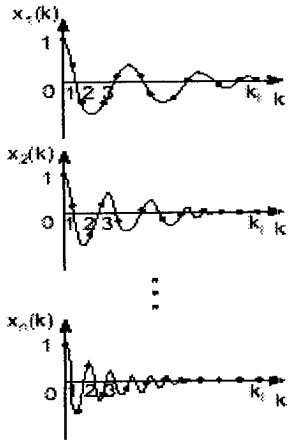


Рисунок 2.7 – Імпульсні характеристики змінних стану

Для моделювання текстур використовують модель (2.24) із спрощеною характеристикою векторів шуму, допускаючи, що відліки шуму не корелюють, тоді $Q(t)=R(t)=\sigma^2\delta(t)$, де σ^2 – дисперсія гаусового шуму, $\delta(t)$ – дельта-функція. В цьому випадку модель (2.24) можна подати як авторегресію із ковзним середнім (AutoRegression and Moving Average – ARMA), дана модель має вигляд

$$u_{i,j} = -\sum_{m,n=0(m,n\neq 0)}^{P,Q} a_{m,n} u_{i-m,j-n} - \sum_{m,n=0}^{L,M} b_{m,n} w_{i-m,j-n}, \qquad (2.26)$$

де $u_{i,j}$ – відліки сигналу зображення, $a_{m,n}$ – коефіцієнти лінійного передбачення (ЛП), $w_{i,j}$ – відліки білого шуму, P,Q – параметри, що задають порядок моделі умовно по координатах OX, OY, $b_{m,n}$, L, M – коефіцієнти та порядок складової ковзного середнього. Модель (2.26) охоплює дві складові – модель авторегресії (AR) (autoregression – AR), що описує динамічні властивості сигналу зображення, та модель ковзного середнього (moving average – MA), що описує властивості сигналу похибки. Для розв'язання задач розпізнавання та класифікації використовуємо саме авторегресійну складову, модель MA похибки використовують як допоміжну, що характеризує якість моделі AR. Тому більш широке застосування в обробці зображень знайшла модель AR

$$u_{i,j} = -\sum_{m,n=0(m,n\neq 0)}^{P,Q} a_{m,n} u_{i-m,j-n} + \varepsilon_{i,j}, \qquad (2.27)$$

де $\varepsilon_{i,j}$ – відліки похибки з дисперсією σ_ε^2.

Параметри моделі (2.27) – коефіцієнти ЛП – можна знайти за допомогою методу найменших квадратів. Одною з найбільш важливих задач при синтезі моделі AR є вибір її порядку таким чином, щоб модель була адекватна даним і, разом з тим, не була чутливою до незначних флуктуацій. Критерієм адекватності моделі може служити співвідношення потужності сигналу зображення до величини похибки (шуму). В літературі це визначають терміном Signal to Noise Ratio (SNR):

$$SNR = 20 log_{10} \frac{P(u)}{\sigma_\varepsilon^2}, \qquad (2.28)$$

де $P(u)$ – потужність сигналу базового фрагмента зображення, що використано для визначення параметрів моделі, σ_ε^2 – дисперсія шуму похибки.

Нечутливість моделі до незначних флуктуацій зображення досягають за допомогою умови інваріантності параметрів моделі до зсуву базового фрагмента відносно координат, в яких задано зображення. Це також важлива властивість моделі і їй приділяють особливу увагу в літературі з обробки зображень. Вона стосується всіх перелічених вище моделей. Для того, щоб модель була інваріантною до зсуву координат і разом з тим була адекватною даним, вона повинна виконувати функції апроксимації та

екстраполяції даних сигналу зображення.

У випадках, коли модель AR неадекватна динаміці даних, використовують дискретну нелінійну модель у вигляді ряду Вольтера. Оскільки така модель в разі двовимірних даних потребує значного числа параметрів, то на практиці застосування знайшли моделі не вище третього порядку нелінійності. Нелінійну модель другого порядку можна подати як

$$u_{i,j} = -\sum_{m,n=0(m,n\neq 0)}^{P,Q} a_{m,n}u_{i-m,j-n} - \sum_{k,l=0(k,l\neq 0)}^{L,M}\sum_{m,n=0(m,n\neq 0)}^{L,M} b_{k,l,m,n}u_{i-k,j-l}u_{i-m,j-n} + \varepsilon_{i,j}, \quad (2.29)$$

де $b_{m,n}$, L, M – коефіцієнти та порядок нелінійної складової. Визначити параметри моделі (2.29) можна за допомогою розширеної системи рівнянь, що охоплює квадратичні складові сигналу. Розв'язати таку систему рівнянь можна за допомогою методу найменших квадратів. Тому нелінійна модель простіша, ніж модель ARMA.

Алгоритм обробки зображення за допомогою моделей такий:

1) за допомогою базового фрагмента зображення визначають параметри моделі та параметри, що характеризують якість моделі (наприклад, статистичні параметри шуму похибки);

2) сигнал зображення обробляють за допомогою моделі і визначають параметри якості;

3) якщо параметри якості відповідають параметрам базового фрагмента, то зображення розпізнано, якщо ні, то зображення не розпізнано.

2.1.5 Методи на основі декомпозиції на власні вектори

В попередніх пунктах розглянуто методи на основі статистичного аналізу зображення в просторовій або спектральній областях та методи на основі синтезу динамічних моделей, що відображають зміну зображення в просторі або часі. Спектральний аналіз і динамічні моделі можуть бути поєднані за допомогою визначення власних значень та власних функцій моделей. Перевагою такого підходу є те, що зображення може характеризуватись мінімальним числом параметрів, наприклад резонансних частот. Причому, відбираються тільки ті резонансні частоти, амплітуда яких найбільша. Відомо декілька підходів до реалізації декомпозиції на власні вектори:

1) синтез фільтрів на основі власних векторів зображення або його кореляційної матриці;

2) відбір принципових (найвпливовіших) компонент розкладання на власні або сингулярні вектори – Principal Component Analysis (PCA);

3) модальний аналіз – Empirical mode decomposition (EMD).

Метод PCA реалізують за допомогою сингулярного розкладання (Singular Value Decomposition – SVD). Перевагою SVD є те, що сингулярне розкладання реального сигналу дає систему ортогональних векторів, також реальних. Ортогональність та те, що дані реальні, спрощує алгебраїчні операції з матрицями.

Аналіз за допомогою розкладання за власними та сингулярними векторами застосовують до сигналу зображення або його кореляційної матриці. Одним з різновидів кореляційної характеристики зображень є розподіл Вігнера.

Недоліком аналізу зображень за допомогою власних і сингулярних векторів є те, що для обчислення векторів потрібні значні обчислювальні ресурси. Більш простим є модальний аналіз. Емпіричні моди визначають за допомогою виділення локальних максимумів і мінімумів сигналу зображення та їх апроксимації поверхнею у вигляді полінома або за допомогою спеціальних функцій. Отримана мода вилучається із зображення і за допомогою сигналу остачі таким же способом визначають наступну моду. Такий ітераційний процес повторюють поки сигнал остачі не стане менше певного граничного значення. В результаті зображення є сумою модальних сигналів та незначної остачі. Моди використовують як характерні ознаки зображення у вигляді масок для класифікації стаціонарних текстур та як кореляційні фільтри для класифікації динамічних текстур.

Алгоритм аналізу зображень за допомогою власних векторів залежить від того, як вони використовуються – як фільтри чи як базис для спектрального аналізу.

2.1.6 Методи класифікації елементів зображень

Оскільки задачі розпізнавання об'єктів полягають у класифікації зображень на основі певних критеріїв, то важливим етапом є вибір оптимального класифікатора. Серед існуючих методів класифікації можна виділити:

- ймовірнісний критерій якості класифікації;
- оптимальна стратегія статистичної класифікації;
- класифікатор Байєса;
- мінімаксний класифікатор;
- класифікатор Неймана-Пірсона.

Класифікатор Байєса є найбільш широкорозповсюдженим з перелічених класифікаторів і застосовується при наявності повної апріорної інформації про класи, тобто коли відомі функція правдоподібності для кожного з класів, матриця штрафів, апріорні ймовірності для кожного з класів. Класифікатор Байєса ґрунтується на основі принципу максимуму апостеріорної ймовірності, що базується на

трьох гіпотезах.

1. Множина $X \times Y$ є ймовірнісним простором з ймовірнісною мірою P. Прецеденти $(x_1, y_1), ..., (x_l, y_l)$ з'являються випадково і незалежно у відповідності з розподілом P.

2. Відомі щільності розподілу класів $p_y(x) = p(x \mid K_y), y \in Y$, що називаються функціями правдоподібності.

3. Відомі ймовірності появи об'єктів кожного з класів $P_y = P(K_y), y \in Y$, що називаються апріорними ймовірностями.

Базуючись на даних гіпотезах, принцип максимуму апостеріорної ймовірності записується у вигляді:

$$F(x) = arg \max_{y \in Y} P(K_y \mid x) = arg \max_{y \in Y} p_y(x) P_y.$$

Доведено, що такий вибір вирішального правила є оптимальним з погляду мінімізації загального ризику. Основна проблема полягає в тому, що на практиці гіпотези 2 і 3 майже ніколи не виконуються. Спроби оцінити ці функції розподілу за навчальною вибіркою могли б привести до деякого результату, якби не погана обумовленість задачі, що приводить до вироджених рішень.

Існують системи виявлення об'єктів зображення, що базуються на «наївному» байєсовому методі. Даний метод ґрунтується на побудові емпіричної щільності розподілу ймовірностей класів за навчальною вибіркою за припущення про незалежність компонентів вектора ознак.

Практичні результати такі:

1. Алгоритм побудови «наївного» байєсового класифікатора схильний до перенавчання;

2. Алгоритм побудови «наївного» байєсового класифікатора чутливий до шуму, тому що базується на емпіричних функціях щільності розподілу;

3. Швидкість роботи самого класифікатора висока, основний час може займати обчислення вектора ознак.

На основі поєднання байесового підходу і теорії графів утворюють байєсові мережі. Суть даного підходу в тому, що будують граф, кожна вершина якого відповідає якому-небудь компоненту вектора ознак, дуги позначають причинно-наслідковий зв'язок. Побудова мережі може бути здійснена автоматично шляхом аналізу кореляції компонентів вектора ознак.

Проблемою для байєсової мережі є погана обумовленість, тому що велика розмірність вектора ознак робить граф зв'язків складним для побудови й аналізу. Також суттєво зростає обчислювальна складність. Одним з варіантів розв'язання даної проблеми є скорочення розмірності вектора ознак, що приводить до погіршення узагальнювальної здатності.

Класифікація за статистичними ознаками можлива після попередньої обробки зображення за допомогою фільтрів, що дозволяє зменшити варіацію параметрів ознак.

Наряду з статистичними методами класифікації в обробці зображень використовують *евристичні методи.* Це ряд підходів, що їх можна розділити на групи.

1. Евристичні методи:

- повна евристична модель, де експертом складається набір правил, що описують зображення об'єкта (будується модель), згідно з якими здійснюється розпізнавання;
- пошук характерних інваріантних ознак, де евристично описуються не зображення шуканого об'єкта в цілому, а його характерні ознаки, інваріантні щодо можливих спотворень (зміна освітленості, поворот, масштабування).

2. Метод порівняння з шаблоном. Складається шаблон для зображення всього об'єкта чи його характерних ознак. Також вводиться функція перевірки відповідності.

3. Методи з навчанням за прецедентами. Модель автоматично будується на основі набору зображень об'єкта, складених заздалегідь з можливих вхідних даних системи.

Евристичні методи є історично найбільш ранніми, вони досить прості в реалізації і працюють з високою швидкістю, однак жорстко запрограмовані правила позбавляють систему гнучкості та стійкості. Як правило, евристичні системи орієнтовані на відносно вузький клас задач.

Метод порівняння з шаблоном є більш універсальним підходом, однак даний метод вимагає наявності дуже точного шаблона об'єкта зображення. Шаблон може бути складною структурою і допускати різні деформації і перетворення, сприяючи, таким чином, інваріантості системи до просторових спотворень об'єкта зображення і змін освітленості. Системи, засновані на порівнянні з шаблоном, найчастіше використовуються для розв'язання задач відстеження об'єктів у відео з ініціалізацією на першому кадрі – до початку роботи системи існує загальна модель шаблона, при ініціалізації вона уточнюється і коректується під час роботи системи. При добре заданому шаблоні досягається висока точність і дуже низький рівень збоїв. Інваріантість до просторових спотворень і зміни освітлення залежить від складності шаблона.

Методи з навчанням за прецедентами є найбільш загальним підходом. Задача розпізнавання об'єктів зображення зводиться до задачі класифікації і для неї застосовується добре розроблений математичний апарат побудови моделі (навчання) за прецедентами. Модель будується автоматично по заздалегідь зібраному наборі прецедентів – зображень, для яких відомо, є вони зображеннями об'єкта чи ні. Спостереженням, у даному випадку, є деякий «вектор ознак», отриманий з вихідного

зображення за допомогою перетворення, що відображає зображення в просторі дійсних векторів. Гіпотеза, що підлягає перевірці, – належність зображення до класу зображень шуканого об'єкта. Таким чином, система розпадається на два модулі – модуль перетворення зображення у вектор ознак і модуль класифікації. Задачею модуля перетворення є найбільш повне й інформативне подання зображення у вигляді числового вектора. Задачею модуля класифікації є перевірка гіпотези належності зображення класу зображень об'єкта на підставі спостереження, яким є вектор ознак. Серед евристичних найбільш поширені методи:

- опорних векторів (SVM, supporting vectors method);
- Sparse network of Winnows (SNoW);
- посилення слабких класифікаторів (classifier boosting).

Метод SVM полягає в побудові оптимального лінійного класифікатора. Класичний алгоритм полягає в побудові лінійної поверхні (гіперплощини), яка рівновіддалена від опуклих оболонок класів, опукла оболонка будується за прецедентами. Стверджується, що така гіперплощина буде оптимальною, з погляду загального ризику, щодо будь-яких інших можливих гіперплощин. Якщо такої гіперплощини не існує (класи лінійно нероздільні), то для здійснення нелінійної класифікації застосовується ядрове перетворення, що проектує вихідний простір у простір ще більшої, можливо нескінченної, розмірності.

Метод опорних векторів був успішно застосований для задачі розпізнавання об'єктів зображення. Метод характеризується:

- високою стійкістю до перенавчання;
- чутливість до шуму може регулюватися за рахунок зменшення точності;
- в системах розпізнавання об'єктів метод дає прискорення в декілька разів у порівнянні з нейронними мережами.

SNoW – особливий вид нейронної мережі. Вектор ознак бінарний. Мережа складається з двох (за числом можливих класів) лінійних нейронів, зв'язаних з компонентами вектора ознак. Класифікація проходить за принципом «переможець забирає все». Метод SNoW вважається досить ефективним методом для розв'язання задач виявлення об'єктів зображення. Відповідно до деяких досліджень SNoW перевершує за своїми параметрами метод опорних векторів.

Classifier boosting – це підхід до розв'язання задачі класифікації шляхом комбінування примітивних класифікаторів в один більш сильний класифікатор. Основна ідея методу полягає в ітеративній мінімізації опуклого функціонала помилки класифікації шляхом додавання в набір класифікаторів чергового слабкого класифікатора. Для систем розпізнавання об'єктів зображення був застосований каскадний підхід, який полягає в побудові каскаду із комплексу слабких класифікаторів, що працює за принципом послідовних наближень. Характеристики

каскадного методу перевершують всі інші системи.

За показниками роботи в реальних системах розпізнавання об'єктів зображення найбільш вдалими виявилися алгоритми boosting (посилення слабких класифікаторів) і SNoW. Обидва підходи забезпечують високу швидкість, високий рівень розпізнавання і низький рівень похибок другого роду. Метод опорних векторів досить сильно поступається за показниками перерахованим вище підходам, тому що має дуже низький відсоток достовірних виявлень.

2.1.7 Методи визначення контурів елементів зображень та сегментації

Для визначення контурів зображень використовують статистичний аналіз фрагментів зображення та їх взаємну кореляцію з метою знаходження стрибкоподібних змін кольору і освітленості. Велика група методів заснована на використанні математичних моделей, що відображають певну взаємодію між окремими пікселями або фрагментами зображень. Також для розв'язання задач розпізнавання об'єктів застосовують різні методи фільтрації, наприклад інверсні фільтри, фільтри Вінера, Байєса. При цьому використовують аналогію між динамікою зображень та фізичними процесами, наприклад дифузії. Для розв'язання деяких задач використовують стохастичні моделі.

Ряд методів визначення контурів зображень реалізовано в середовищі Matlab. У випадку динамічного фону дані методи не дозволяють отримати інформативний результат, оскільки перетворюють зображення в безліч контурних об'єктів. В цьому можна переконатись, здійснивши розпізнавання контурів зображення за допомогою функції edge, вибравши один із запропонованих фільтрів Собеля, Превіта, Робертса, лапласіан-гаусіана або за методом Канні.

Розпізнавання об'єктів за допомогою визначення контурів виконується функцією *edge*. Суть методу полягає в пошуку локальних ділянок з перепадами яскравості. Перепади яскравості шукають за допомогою фільтрації по кожній з осей координат одновимірним фільтром лапласіан-гаусіана. У методі Канні для класифікації перепадів на «слабкі» і «сильні» використовується два пороги – нижній і верхній. «Слабкі» границі відзначаються в результуючому зображенні, тільки якщо вони з'єднані з «сильними». Для зображень, що зіпсовані шумом, даний метод забезпечує найкраще виявлення границь у порівнянні з іншими методами функції *edge* та методом сегментації, але вимагає істотно більшого часу.

Найкращий результат отримано за методом Канні, що наведено на рис. 2.8.

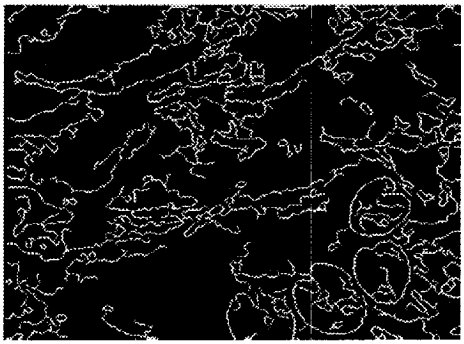


Рисунок 2.8 – Розпізнавання об'єктів за допомогою функції edge

Алгоритми сегментації зображень базуються на одній з двох характеристик сигналу яскравості – розривності або однорідності. В першому випадку підхід базується на розбитті зображення на основі різких змін сигналу, таких як перепади яскравості на зображенні. Зазвичай пошук розривів здійснюється за допомогою ковзних масок. Друга категорія методів базується на визначенні однорідності зображення згідно з наперед обраними критеріями. Прикладами таких методів є порогова обробка, злиття та розбиття областей. За допомогою Mathlab сегментацію можна виконати застосовуючи функцію *qtdecomp* за методом розподілу. Суть методу така. Зображення розбивається на блоки, що не перекриваються. Кожний блок за допомогою деякого критерію перевіряється на однорідність. Якщо блок неоднорідний, то він розбивається на блоки меншого розміру. Процес завершується тоді, коли жодний з блоків не може бути розділений.

2.2 Фрактальні методи

У той час як об'єкти, побудовані людиною, такі як промислові та житлові будинки, можуть бути ефективно описані набором простих геометричних примітивів: кубів, сфер, циліндрів, конусів, кольорові текстури природного походження, через свою нерегулярність і фрагментарність, погано піддаються такому опису. У зв'язку з цим, для аналізу таких текстур виявляється природним подання їх у вигляді фрактала з деяким розміром D.

Фрактал (лат. *fractus,* fractal – дроблений) – термін, який ввів Бенуа Мандельброт в 1975 році для позначення нерегулярних самоподібних множин.

Фрактал – це нескінченно самоподібна геометрична фігура, кожний фрагмент якої повторюється при зменшенні масштабу. Масштабна

інваріантність, що спостерігається у фракталах, може бути або точною, або наближеною.

Ще один варіант визначення: **фрактал** – самоподібна множина нецілої розмірності. Самоподібна множина – множина, що подається у вигляді об'єднання однакових непересічних підмножин подібних до вихідної множини.

Основні властивості фракталів.

- Вони мають тонку структуру, тобто містять довільно малі масштаби.
- Вони занадто нерегулярні, щоб бути описаними традиційною геометричною мовою.
- Вони мають деяку форму самоподоби, допускаючи наближену.
- Вони мають дробову «фрактальну» розмірність, що її також називають розмірністю Мінковського.

У наш час фрактали знайшли своє застосування при аналізі текстур ландшафтів, отриманих при аерокосмічній зйомці, при аналізі поверхонь порошків та інших пористих середовищ, при аналізі поверхні хмар тощо. Однак розмір фрактала кольорової текстури багато в чому залежить від вибору методу оцінювання. Так, при використанні різних методів оцінювання розміру фрактала ми одержимо відповідно й різні його розміри. Зіставлення текстур, таким чином, можливо при використанні того самого методу (групи методів). Більше того, не всі текстури добре розрізняються за розміром фрактала. У зв'язку з цим перш, ніж вносити в систему ознак розмір фрактала, необхідно оцінити фрактальність текстури. Оцінювання фрактальності текстури здійснюється на основі обраного методу оцінювання розміру фрактала. Оскільки розмір фрактала обчислюється через оцінку вибіркової регресії, то природно оцінювати фрактальність текстури за коефіцієнтом кореляції між логарифмом випадкової величини й логарифмом заданої функції кроку. При цьому ухвалення рішення про фрактальність текстури можна будувати таким чином:

1. Побудувати залежність коефіцієнта кореляції від кроку; значення кроку, при якому функція має максимум, є максимальним кроком у задаваному діапазоні кроків при оцінюванні розміру фрактала;
2. Не враховувати оцінку розміру фрактала при низькому коефіцієнті кореляції в тих методах, де використовується оцінка фрактала як середнє значення в серії експериментів;
3. Не вносити розмір фрактала в систему ознак для сегментації текстур при значеннях коефіцієнта кореляції $< 0,7$.

Оцінка фрактальності текстури є важливою характеристикою при сегментації за розміром фрактала.

Поширення фрактального опису пояснюється тим, що більшість просторових систем у природі є нерегулярним і фрагментарним, форма цих систем погано піддається опису апаратом евклідової геометрії.

Наприклад, берегова лінія острова не пряма й не кругла, і ніяка інша класична крива не може служити для опису й пояснення її форми без надмірної штучності й ускладнення.

Фрактальні структури є одним із різновидів текстур, де деталізація зображення досягається поданням об'єкта подібними меншої величини. Деякі динамічні текстури, наприклад поверхні лісу і хвиль води, хмари, пористі мінерали, металоструктури, можуть бути подані за допомогою фрактальної екстраполяції. Суть фрактальних методів в задачах розпізнавання об'єктів викладена нижче. Обриси штучних об'єктів – танків, автомобілів, створюються лініями, що описуються рівняннями цілого порядку. Природні об'єкти – рельєф, дерева – фрактальні, тобто мають фрактальну розмірність. Використовуючи цю властивість за допомогою фрактальної апроксимації об'єктів зображення можна створити систему розпізнавання образів. Така система не бачить кущ, але добре розпізнає штучний об'єкт, схований за кущем. Основною перевагою даного методу над іншими є те, що він не чутливий до завад. На результат розпізнавання не впливає колір та контрастність об'єкта відносно фону, впливає лише площа, яку займає шуканий об'єкт на зображенні. В цьому полягає недолік даного методу – за допомогою фрактального аналізу можна розпізнавати об'єкти, що за розміром порівнянні зі структурою елементів фону.

Класифікації фракталів. В основному фрактали ділять на геометричні, алгебраїчні й стохастичні. Однак існують і інші класифікації: рукотворні й природні. До рукотворних належать ті фрактали, які були винайдені вченими, вони при будь-якому масштабі мають фрактальні властивості. На природні фрактали накладається обмеження на область існування – тобто максимальний і мінімальний розмір, при яких у об'єкта спостерігаються фрактальні властивості.

Для побудови алгебраїчних фракталів (рис. 2.9) використовуються ітерації нелінійних відображень, що задаються простими алгебраїчними формулами. Найбільш вивчений двовимірний випадок. Нелінійні динамічні системи можуть мати декілька стійких станів. Кожний стійкий стан (атрактор) має деяку ділянку початкових станів, при яких система обов'язково в нього перейде. Таким чином, фазовий простір розбивається на ділянки притягання атракторів.

Якщо фазовим є двовимірний простір, то, зафарбовуючи ділянки притягання різними кольорами, можна одержати кольоровий фазовий портрет цієї системи (ітераційного процесу). Змінюючи алгоритм вибору кольору, можна одержати складні фрактальні картини з вигадливими багатобарвними візерунками. Несподіванкою для математиків стала можливість за допомогою примітивних алгоритмів породжувати дуже складні нетривіальні структури. Приклади алгебраїчних фракталів: множина Мандельброта, множина Жюліа, басейни Ньютона, біоморфи.

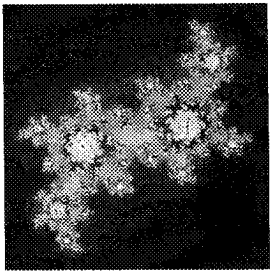


Рисунок 2.9 – Приклад алгебраїчного фрактала. Множина Жюліа

Геометричні фрактали (рис. 2.10) застосовуються для одержання зображень дерев, кущів, берегових ліній тощо. Алгебраїчні та стохастичні – при побудові ландшафтів, поверхні морів, моделей біологічних об'єктів та інше.

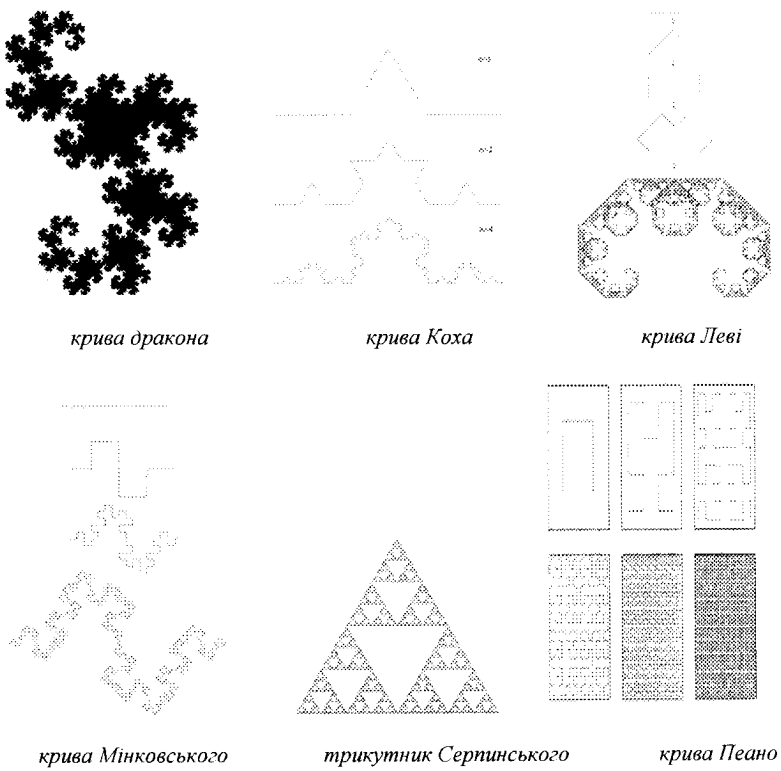


Рисунок 2.10 – Приклади геометричних фракталів

З математичної точки зору, фрактал – це, передусім, множина з дробовою розмірністю (fractional dimension). Ми добре уявляємо собі, що точка має розмірність 0, коло та відрізок – розмірність 1, куб та сфера – 2. З одновимірними об'єктами ми пов'язуємо поняття довжини, з двовимірними – площі і т. д. Але як можна уявити собі множину з розмірністю 3/2? Для цього необхідно дещо проміжне між довжиною та площиною, і якщо довжину умовно назвати 1-вимірною, а площу – 2-вимірною, то необхідний 3/2-вимір. Хаусдорф визначив такий α-вимір для будь-якої $\alpha \geq 0$ і на цій основі кожній множині в евклідовому просторі надав у відповідність число. Для пояснення фрактальної розмірності необхідно ввести поняття топологічної розмірності. Під топологічною розмірністю множини в лінійному просторі розуміють число лінійно незалежних координат в просторі. Фрактальна розмірність множини – розмірність того простору, який повністю заповнюється множиною. Для зв'язку фрактальної та топологічної розмірності використовують показник Херста H, який обчислюється за формулою $H = D - Dt$. Ідеї Хаусдорфа були розвинуті А. С. Безіковичем. В наступні роки розмірність Хаусдорфа-Безіковича отримала застосування в деяких розділах математики, але нічого не передбачувало їй тієї популярності цього поняття за межами математики, яка спостерігається тепер. Частково цьому допомогла наукова діяльність Б. Мандельброта, який в своїх книгах навів яскраві приклади застосування фракталів для пояснення деяких природних явищ. Тобто, фрактальна розмірність, як правило, є невід'ємним нецілим числом, яке показує деяким чином геометричну складність об'єкта.

Розмірність фрактала D визначається як

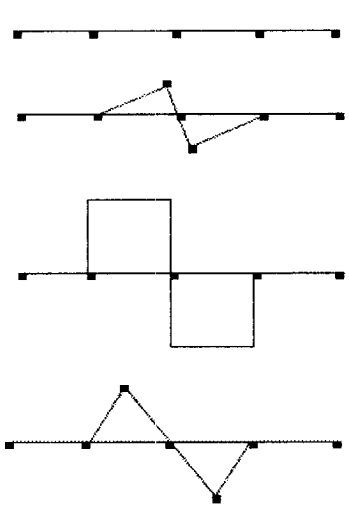
$$D = \frac{\log N}{\log(1/r)}, \tag{2.30}$$

де $1/r$ – співвідношення подібності, N – число кроків, необхідне для того, щоб покрити криву.

Практично розмір фрактала для кривої оцінюється шляхом вимірювання довжин кривої при різних розмірах кроку. Розмірність фрактала D може бути оцінена за допомогою такого рівняння регресії:

$$\begin{aligned} \log L &= C + B \log G \\ D &= 1 - B \end{aligned}, \tag{2.31}$$

де L – довжина кривої, B – нахил регресії, G – величина кроку, C – константа.



$N = 4, r = 1/4$

$D = \log 4 / \log 4 = 1$

$N = 5, r = 1/4$

$D = \log 5 / \log 4 = 1.16$

$N = 8, r = 1/4$

$D = \log 8 / \log 4 = 1.5$

$N = 6, r = 1/4$

$D = \log 6 / \log 4 = 1.28$

Рисунок 2.11 – Визначення розміру фрактала на прикладі чотирьох ламаних ліній

Розглянемо докладніше реалізацію фрактального підходу до аналізу хмар. В основу цього методу покладено виведене Мандельбротом співвідношення між периметром і площею об'єкта. Для кіл, квадратів, рівносторонніх трикутників та інших багатокутників відношення периметра до квадратного кореня з площі, що ним обмежується, не залежить від розміру фігури і є постійною величиною для даної сім'ї. Аналогічно для сім'ї подібних островів відношення довжини нефрактальної берегової лінії будь-якого острова до квадратного кореня з його площі не залежить від розміру площі. Однак, якщо берегова лінія фрактальна, то її довжина $L(\delta)$ залежить від довжини еталона δ і прямує до нескінченності якщо еталон також прямує до нуля.

При цьому площа острова $A(\delta)$, обумовлена кількістю квадратів δ^2, що на ній розташовані, залишається кінцевою. Таким чином, відношення периметра до квадратного кореня із площі розходяться. Мандельброт для випадку фрактальной берегової лінії одержав таке співвідношення між периметром і площею:

$$L(\delta) = C\delta^{(1-D)}[A(\delta)]^{D/2}. \qquad (2.32)$$

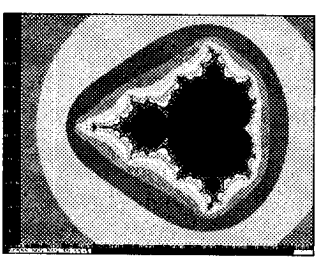
Це співвідношення виконується для будь-якого еталона довжини δ, досить малого, щоб задовільно виміряти найменший з островів.

Співвідношення (2.32) застосовується при дослідженні геометрії хмар і зон дощу, розміри яких знаходяться в широких межах від 1 до $1{,}2\times10^6$. З'ясувалося, що периметр хмари пов'язаний з його площею співвідношенням (2.32) з фрактальним розміром D = 1,35 ± 0,05. При цьому ці оцінки виявилися справедливі як для купчастих, так і для пір'ястих хмар. У роботі А. Вальдфогеля, присвяченій аналізу фрактальної розмірності хмар з потужними конвективними струмами, було встановлене співвідношення між периметром і площею для послідовності моментів часу (з інтервалом в 1 хвилину) у площині перерізу для постійного коефіцієнта відбиття. Основні висновки можуть бути такими: для хмар, периметр яких більше 8 км, розмір фрактала приблизно збігається з розміром менш потужних хмар і становить 1,36 ± 0,1; для хмар з периметром від 3 км до 8 км – D =1,0 ± 0,1 і, нарешті, хмари з периметром менш 3 км не є фракталами.

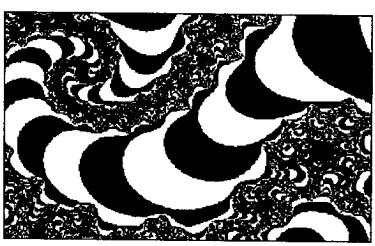
Мандельброт запропонував не тільки означення фракталів, але також і алгоритм побудови одного з них, що отримав назву на честь ученого. Алгоритм побудови множини Мандельброта заснований на ітеративному обчисленні за формулою:

$$Z[i+1]=Z[i]\cdot Z[i]+C,$$

де Z і C – комплексні змінні. Ітерації виконуються для кожної стартової точки C прямокутної або квадратної області – підмножини комплексної площини. Ітераційний процес триває доти, поки $Z[i]$ не вийде за межі кола заданого радіуса, центр якого лежить у точці (0,0), або після досить великої кількості ітерацій. Залежно від кількості ітерацій, протягом яких $Z[i]$ залишається всередині кола, встановлюється колір точки C. Якщо $Z[i]$ залишається всередині кола протягом досить великої кількості ітерацій, то ця точка растра зафарбовується в чорний колір. Множині Мандельброта (рис. 2.12) належать саме ті точки, які протягом нескінченного числа ітерацій не переходять у нескінченність.



а)



б)

Рисунок 2.12 – а) Множина Мандельброта; б) збільшена ділянка границі множини Мандельброта

Побудова іншої фрактальної множини, сніжинки Коха (рис. 2.13), починається з правильного трикутника, довжина сторони якого дорівнює 1. Сторона трикутника вважається базовою ланкою для вихідного положення. Далі, на будь-якому кроці ітерації кожна ланка заміняється на утворювальний елемент – ламану, що складається по краях з відрізків довжиною 1/3 від довжини ланки, між якими розміщаються дві сторони правильного трикутника зі стороною в 1/3 довжини ланки. Всі відрізки – сторони отриманої кривої – вважаються базовими ланками для наступної ітерації. Крива, що одержується в результаті n-ї ітерації при будь-якому кінцевому n, називається передфракталом, і лише при n, що наближається до нескінченності, крива Коха стає фракталом. Отримана в результаті ітераційного процесу фрактальна множина є лінію нескінченної довжини, що обмежує кінцеву площу. Дійсно, при кожному кроці число сторін результувального багатокутника збільшується в 4 рази, а довжина кожної сторони зменшується тільки в 3 рази, тобто довжина багатокутника на n-ій ітерації дорівнює $3\cdot(4/3)^n$ і прагне до нескінченності з ростом n.

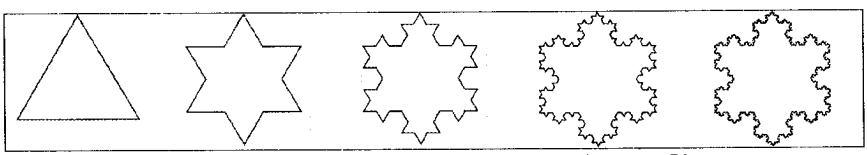


Рисунок 2.13 – Перші 5 поколінь сніжинки Коха

Площа під кривою, якщо прийняти площу утворювального трикутника за 1, дорівнює:

$$S = 1 + 1/3 \sum_{k=0}^{\infty} (4/9)^k = 1,6.$$

З недавнього часу фрактальні методи почали використовувати при розробці методів розпізнавання образів на радіолокаційних зображеннях. Суть їх така. Важко локалізувати танк, замаскований серед кущів. Важко, навіть коли є якісний сигнал від теле- та теплотелевізора. Набагато легше зробити це за допомогою фрактальних методів. Як вже було сказано вище, обриси штучних об'єктів – танків, автомобілів – створені лініями, які описуються рівняннями цілого порядку. А ось об'єкти природні – рельєф, дерева – фрактальні, тобто мають фрактальну розмірність. Ось на цьому принципі і побудовані нові системи розпізнавання образів. Системи розпізнавання не бачать кущ, але дуже добре розпізнають штучний об'єкт, схований за кущом. Маскувальне забарвлення може допомогти, але якщо воно не створене кривими другого порядку, як звичайно.

Іншими словами, якщо ми виміряємо розмірність зображення якогось природного ландшафту, то вона буде дробова. Розмірність геометричної фігури рівна близько 2 (через похибку вимірювання). А коли накласти, наприклад, прямокутник (як це показано на рисунку 2.14) на природне зображення, то розмірність всієї картинки різко поміняється.

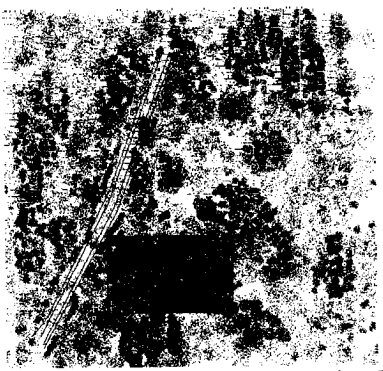


Рисунок 2.14 – Розрахунок розмірності зображення

Основною перевагою даного методу над іншими є те, що не потрібно витрачати зусилля і час на покращення якості зображення. Це не дуже впливає на результат. Інша перевага полягає в нижчій вимозі до високої роздільної здатності зображень, порівняно з іншими методами. На результат впливає лише площа, яку займає штучний об'єкт на зображенні, а не контраст, як звичайно.

Самоподібність (self-similarity) є основною характеристикою фрактала і означає, що він більш-менш одноманітно побудований у широкому діапазоні масштабів. Так, при збільшенні маленькі фрагменти фрактала виходять дуже схожими на більші. В ідеальному випадку така самоподібність приводить до того, що фрактальний об'єкт є інваріантним до збільшень.

Звичайно, для реального природного фрактала існує деякий мінімальний масштаб довжини $l_{\min}$ такий, що на відстанях $l \approx l_{\min}$ його основна властивість – самоподібність – пропадає. Крім того, на досить великих масштабах довжин $l > l_{\max}$, де $l_{\max}$ – характерний геометричний розмір об'єктів, ця властивість самоподібності також порушується. Тому властивості природних фракталів розглядаються лише в масштабах l, що задовольняє співвідношення $l_{\min} \le l \le l_{\max}$.

Відмітимо, що властивість точної самоподібності характерна лише для регулярних фракталів. Якщо замість детермінованого способу побудови внести в алгоритм їхнього створення деякий елемент випадковості (як це

буває, наприклад, у багатьох процесах дифузійного росту кластерів, електричному пробої й т. д.), то виникають так звані випадкові фрактали. Основна їхня відмінність від регулярних полягає в тому, що властивості самоподібності справедливі тільки після відповідного усереднення по всіх статистично незалежних реалізаціях об'єкта. При цьому збільшена частина фрактала не точно ідентична вихідному фрагменту, однак їхні статистичні характеристики збігаються.

Стиск зображень (image compression). За допомогою фракталів можна стискувати зображення з деякою втратою якості аналогічно іншим методам стику з втратами. Але фрактальний стиск дає кращі результати. Методи компресії, основані на RLE, класичний алгоритм Хаффмана, LZW не враховують природи стискуваних даних і тому дають незадовільні результати при обробці зображень. Фрактальний стиск зображень – це алгоритм стиску зображень з втратами, заснований на застосуванні систем IFS до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє одержати дуже високі коефіцієнти стиску (кращі приклади – до 1000 разів при прийнятній візуальній якості) для реальних фотографій природних об'єктів, що недоступно для інших алгоритмів стиску зображень у принципі.

Основна проблема фрактального стиску – це те, що компресія-декомпресія виконується швидко і однозначно в той час, як пряма процедура потребує від машини великих інтелектуальних можливостей. При компресії можна не зберігати оригінальні розміри зображення, достатньо просто запам'ятати їх відношення. А при декомпресії – задавати ті розміри, які нам найбільше підходять. Така можливість дозволяє вирішити задачу екстраполяції початкового зображення. При встановленні нових розмірів, що перевищують старі, в нове зображення додаються елементи, подібні іншим елементам зображення. І якщо обробляється природний об'єкт (наприклад, гранітний камінь), то заміна не буде помітною.

Основа методу фрактального кодування – це виявлення самоподібних ділянок у зображенні. Патенти ідеї були отримані в 1990-1991 роках.

Суть фрактального стиску. В основі більшості методів фрактального кодування, що застосовуються сьогодні, використовуються системи доменних і рангових блоків зображення, блоків квадратної форми, що покривають все зображення. Фрактальне кодування напівтонових зображень основане на гіпотезі, згідно з якою в будь-якому зображенні можна знайти локальну самоподібність різних його частин. Існуючі алгоритми фрактального стиску, як правило, притримуються такої схеми кодування. Зображення, яке кодується розбивається на множину блоків, що не перекриваються (рангові області), для кожного з яких, в межах цього ж зображення, відшукується блок більшого розміру (домен), пікселі якого, шляхом деякого перетворення, переводились би в пікселі рангової області.

При цьому для пошуку оптимальної відповідності рангових областей і доменів необхідний повний перебір варіантів, що веде за собою значні обчислювальні затрати. З перетворень, що переводять домени в рангові області, формується відображення, що переводить зображення в зображення. При цьому кодом зображення буде місце розташування і розміри рангових областей, а також коефіцієнти перетворень, які описують самоподібність всередині зображення. Кількість бітів, необхідних для опису коду, буде значно менше кількості бітів, необхідних для опису початкового зображення. *Коефіцієнтом стиску* називається відношення бітового подання зображення до бітового подання коду. В відомих фрактальних методах стиску зображень значення цього коефіцієнта може досягати 100 при достатньо непоганій якості відновлення. Для відновлення закодованого таким чином зображення використовується принцип стиснених відображень, який говорить, що стискувальне відображення, що діє в повному метричному просторі, має єдину нерухому точку. Відображення, що діє на повному метричному просторі зображень, формується з перетворень, які переводять домени в рангові області [3].

Відповідно до даного методу зображення розбивається на безліч неперекривних рангових підзображень і визначається безліч перекривних доменних підзображень. Для кожного рангового блоку алгоритм кодування знаходить найбільш підходящий доменний блок і афінне перетворення, що переводить цей доменний блок у даний ранговий блок. Структура зображення відображається в систему рангових блоків, доменних блоків і перетворень.

Основна складність фрактального стиску полягає в тому, що для знаходження відповідних доменних блоків, загалом кажучи, потрібен повний перебір. Оскільки при цьому переборі щораз повинні порівнюватися два масиви, дана операція виходить досить тривалою. Порівняно простим перетворенням її можна звести до операції скалярного добутку двох масивів, однак навіть скалярний добуток обчислюється порівняно тривалий час.

Крім стиску, іншою областю фрактальної обробки зображень є їх генерація. В наш час існує множина найрізноманітніших пакетів прикладних програм (від простих, які створюють зображення на основі множини Мандельброта (Fractal SSE), до складних, які генерують зображення 3d, анімаційні зображення та IFS-зображення). Всі вони побудовані на основі відкриття Мандельброта: якщо нанести визначені точки на площину комплексних чисел, то можна створювати зображення надзвичайного абстрактного вигляду – множина Мандельброта. В рівняння Мандельброта підставляються координати деякої точки комплексної площини, і результатом є координати іншої точки. Результат, отриманий при введенні координат першої точки, слугує початком для наступної ітерації, її результат підставляється в наступне рівняння і так

далі. Обидві ці найголовніші області застосування фрактальних методів в наш час знаходяться на порівняно високому рівні розвитку, незважаючи на те, що фрактальна наука досить молода. Існує надзвичайно велика кількість програм, за допомогою яких можна створити або стиснути зображення, і ефективність обробки зображень деяких з цих програм достатньо висока.

Потенційним, хоч і менш відомим видом фракталів, є фрактал на основі **системи ітераційних функцій (Iterated Function System – IFS).** Метод IFS, який застосовується до побудови фрактальних зображень, винайшов Майкл Барнслі. Він базується на самоподібності елементів зображення і полягає в моделюванні малюнка декількома меншими частинами його самого. Найвідомішим IFS-зображенням є чорний папоротник, в якому кожен лист в дійсності являє собою мініатюрний варіант самого папоротника [4].

Система IFS – це також сукупність стискальних афінних перетворень. Як відомо, афінні перетворення містять у собі масштабування, поворот і паралельний перенос. Афінне перетворення вважається стискальним, якщо коефіцієнт масштабування менше одиниці.

Розглянемо докладніше побудову кривої Коха з використанням афінних перетворень. Кожний новий елемент кривої містить чотири ланки, отриманих з утворювального елемента з використанням масштабування, повороту й переносу.

1. Для одержання першої ланки досить стиснути вихідний відрізок у три рази. Слід зазначити, що те ж масштабування застосовується для всіх ланок.

2. Наступна ланка будується з використанням всіх можливих перетворень, а саме: стиск у три рази, поворот на 60 градусів і паралельний перенос на 1/3 по осі X.

3. Третя ланка будується аналогічно другій: стиск у три рази, поворот на 60 градусів, паралельний перенос на 2/3 вздовж осі X.

4. Остання ланка: стиск у три рази, паралельний перенос на 2/3 вздовж осі X.

Для синтезу фрактала вибирається початкова точка, до якої застосовується випадковим образом обране з IFS перетворення, у результаті чого точка переміщується в інший кінець екрана. Ця операція повторюється багато разів (досить 100 ітерацій), і через деякий час точка починає блукати атрактором (безліч всіх можливих траєкторій), що і буде являти собою зображення фрактала. Кожне нове положення точки зафарбовується кольором, відмінним від фону. Існує теорема, яка доводить, що отриманий атрактор буде замкнутим. Для того, щоб блукаюча точка зафарбовувала нові пікселі, а не блукала старими, використовують сьомий параметр, що являє собою ймовірність появи конкретного афінного перетворення з набору перетворень IFS. Якщо

вибрати початкову точку так, щоб вона відразу виявилася на атракторі, то вона починає блукати в області цього атрактора, не переміщуючись в інші області екрана. Розглядаючи кожне перетворення окремо, можемо помітити, що де б ми не починали, після декількох ітерацій точка перестане рухатися екраном. Точка зупинки називається нерухомою точкою – це розв'язок системи лінійних рівнянь двох змінних, який знаходиться методом простої ітерації. Нерухома точка кожного перетворення входить до складу атрактора. Тому за початкову точку при побудові фрактала можна взяти нерухому точку першого перетворення з набору IFS.

Розмірність. Види розмірності

Ключовою величиною що описує фрактал кількісно є «фрактальна розмірність», однак, у різних джерелах під цим терміном розуміють різні величини: розмірність Мінковського, розмірність Хаусдорфа-Безиковича, розмірність самоподібності. Ці величини відрізняються алгоритмом обчислення, але для математичних фракталів є еквівалентними.

Евклідова або вкладена розмірність D_E (embedding dimension) – це розмірність, в якій об'єкти є набором точок, поміщених в простір. Іншими словами D_E – розмірність простору, що містить об'єкт.

Топологічна розмірність D_T – це ціла величина, що характеризує топологічний об'єкт: для лінії $D_T = 1$ для площини – 2, для поверхні – 3. D_T гомеоморфна, тобто є інваріантом відносно лінійних перетворень.

Розмірність Хаусдорфа-Безиковича D_H – це міра розбиття об'єкта E на частини розміром r з наступним підрахунком числа $N(r)$ частин, що покривають досліджуваний об'єкт.

Розмірність D_H інваріантна щодо лінійних перетворень і для її кількісного оцінювання використовується величина, яка називається метричним порядком:

$$k(E) = -\lim \frac{\ln N(r)}{\ln r}. \qquad (2.33)$$

$k(E)$ пов'язана з розмірністю Хаусдорфа-Безиковича співвідношенням:

$$D_H > k(E). \qquad (2.34)$$

Розмірність самоподібності D_s – характеризує регулярні фрактали (крива Коха, трикутник Серпінського й т. д.) і є для них оцінкою розмірності Хаусдорфа-Безиковича. Значення D_s обчислюється точно, тому що можна чітко виділити компоненти подібності й визначити їх масштабний коефіцієнт. Для обчислення D_s використовується співвідношення:

$$N = r^{D_S}, \tag{2.35}$$

де N – число компонентів подібності, r – масштабний коефіцієнт.

Таким чином, розмірність самоподібності характеризується співвідношенням:

$$D_S = \lim \frac{\ln N(r)}{\ln(r)}. \tag{2.36}$$

Фракталом в широкому сенсі називається множина, у якої розмірність Хаусдорфа-Безиковича D_H не збігається з його топологічною розмірністю D_T.

2.3 Вейвлет-перетворення

Вейвлет-перетворення (wavelet transformation) – це сучасний і перспективний метод обробки даних. Англійське слово wavelet (від французького «ondelette») дослівно перекладається як «коротка (маленька) хвиля». Апарат вейвлет-аналізу одержав свій розвиток на початку 1980-х років у роботах Морле, Гроссмана й деяких інших авторів. Результаты, отримані у різних областях за допомогою вейвлет-аналізу, підсилили інтерес до цього напрямку та сприяють його безупинному розвитку.

Методи вейвлет-аналізу можна застосувати до даних різної природи. Це можуть бути, наприклад, одновимірні функції або двовимірні зображення. Грубу класифікацію вейвлет-алгоритмів можна зробити, виділивши безперервне (CWT – Continuous Wavelet Transform) і дискретне (DWT – вDiscrete Wavelet Transform) вейвлет-перетворення. Одержати набір вейвлет-коефіцієнтів у випадку дискретного перетворення швидше, і воно дає досить точне подання сигналу при меншому обсязі одержуваних у результаті даних. Безперервне перетворення вимагає більших обчислювальних витрат, але, разом із цим, дозволяє детальніше роздивитися структуру сигналу.

На відміну від звичайних спектральних перетворень, вейвлет-аналіз дозволяє з однаковою точністю апроксимувати як гладкі функції, так і функції з різкими випадами, що дає можливість визначати незначні об'єкти. Застосування вейвлетів розглянуто в багатьох роботах. Оскільки вейвлет-перетворення являє собою згортку сигналу зображення з вейвлет-функцією, що зміщується відносно координат, даний метод близький за змістом до методів на основі фільтрації.

Вибір того чи іншого методу залежить від поставленого завдання й типу наявних даних, які необхідно обробити, від можливостей

обчислювальної техніки і від того, у якому вигляді необхідно подати результат.

Термін вейвлет-перетворення (ВП) об'єднує два види перетворень – пряме і обернене, які, відповідно, переводять досліджувану функцію $f(x)$ в набір вейвлет-коефіцієнтів $W_\psi(a,b)f$ і назад. Розділяють безперервне та дискретне перетворення

Пряме вейвлет-перетворення здійснюється відповідно до правила

$$W_\psi(a,b)f = \frac{1}{\sqrt{C_\psi}}\int \frac{1}{\sqrt{|a|}}\psi\left(\frac{x-b}{a}\right)f(x)dx, \qquad (2.37)$$

де a і b – параметри, що визначають відповідно масштаб і зсув функції ψ, яка називається аналізуючим вейвлетом, C_ψ – нормувальний множник. Інтегрування ведуть по всій числовій осі.

Базисний, або материнський вейвлет (parent wavelet) ψ створює за допомогою розтягнень та зсувів сім'ю $\psi\left(\frac{x-b}{a}\right)$.

Маючи відомий набір коефіцієнтів $W_\psi(a,b)f$, можна відновити первинний вигляд функції $f(x)$:

$$f(x) = \frac{1}{\sqrt{C_\psi}}\int\int \frac{1}{\sqrt{|a|}}\psi\left(\frac{x-b}{a}\right)\left[W_\psi(a,b)f\right]\frac{dadb}{a^2}. \qquad (2.38)$$

Пряме (2.37) і обернене (2.38) перетворення залежать від деякої функції $\psi(x)\in L^2(R)$, яку називають базисним вейвлетом. Практично єдиним обмеженням на його вибір є умова скінченності нормувального множника

$$C_\psi = \int_{-\infty}^{\infty}\frac{\left|\hat{\Psi}(\omega)\right|^2}{|\omega|}d\omega = 2\int_0^{\infty}\frac{\left|\hat{\Psi}(\omega)\right|^2}{\omega}d\omega < \infty, \qquad (2.39)$$

де $\hat{\Psi}(\omega)$ – Фур'є-образ вейвлета $\psi(x)$: $\hat{\Psi}(\omega) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\psi(x)e^{-i\omega X}dx$.

Безліч функцій задовольняють дану умову, тому можна підібрати вид вейвлета, який найбільш підходить для рішення конкретного завдання.

Умова (2.37) означає, що Фур'є-образ вейвлета дорівнює нулю при нульовій частоті, тобто $\hat{\Psi}(\omega)\big|_{\omega=0} = 0$. Якщо це не так, то знаменник в

інтегралі (2.37) прямує до нуля, у той час як чисельник має відмінне від нуля значення, і коефіцієнт C_ψ перестає бути кінцевим.

У свою чергу, цю вимогу можна подати в іншому вигляді. Оскільки Фур'є-образ $\hat{\Psi}(\omega)$ при нульовій частоті має вигляд $\int_{-\infty}^{\infty}\psi(x)dx$, ми можемо вимагати рівність нулю інтеграла від вейвлета по всій осі:

$$\int_{-\infty}^{\infty}\psi(x)dx = 0. \tag{2.40}$$

Головні ознаки вейвлета

Як базисні функції, що утворюють ортогональний базис, можна використовувати широкий набір вейвлетів. Для практичного застосування важливо знати ознаки, якими неодмінно повинна володіти вихідна функція, щоб стати вейвлетом. Наведемо основні з них.

Обмеженість. Квадрат норми функції повинен бути скінченним:

$$\|\psi\|^2 = \int_{-\infty}^{\infty}|\psi(x)|^2 dx < \infty \ . \tag{2.41}$$

Локалізація. ВП на відміну від перетворення Фур'є використовує локалізовану вихідну функцію і у часі, і за частотою. Для цього досить, щоб виконувалися умови:

$$|\psi(x)| \le C(1+|t|)^{-1-\varepsilon} \text{ і } |S_\psi(\omega)| \le C(1+|\omega|)^{-1-\varepsilon}, \text{ при } \varepsilon > 0 \ . \tag{2.42}$$

Нульове середнє. Графік вихідної функції повинен бути знакозмінним в околі нуля на осі часу і мати нульову площу

$$\int_{-\infty}^{\infty}\psi(t)dt = 0. \tag{2.43}$$

Рівність нулю площі функції $\psi(t)$, тобто нульового моменту, призводить до того, що Фур'є-перетворення $S_\psi(\omega)$ цієї функції дорівнює нулю при $\omega = 0$ і має вигляд смугового фільтра.

Автомодальність. Характерною ознакою ВП є його самоподібність. Всі вейвлети конкретної сім'ї $\psi_{ab}(t)$ мають те ж число осциляцій, що й

материнський вейвлет $\psi(t)$, оскільки отримані з нього за допомогою масштабних перетворень *(a)* і зсуву *(b)*.

Найбільш поширені базиси створюються на основі похідних функції Гаусса $(g_0(t) = \exp(-t^2/2))$. Це обумовлено тим, що функція Гаусса має найкращі показники локалізації як у часовій, так і в частотній областях.

На рис. 2.15 показані вейвлети перших чотирьох порядків і модулі їх спектральної щільності. При n =1 одержуємо вейвлет першого порядку, який називається WAVE-вейвлетом з нульовим моментом рівним нулю. При n = 2 одержуємо MHAT-вейвлет, що називається «мексиканський капелюх» (mexican hat – схожий на сомбреро). У нього нульовий і перший моменти дорівнюють нулю.

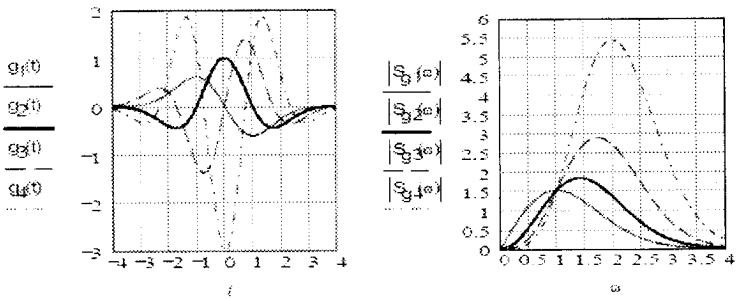


Рисунок 2.15 – Вейвлети перших чотирьох порядків

Найбільш простий приклад дискретного вейвлета – це HAAR-вейвлет. Його недоліком є несиметричність форми та не гладкість – різкі границі в t-області. Серед комплексних вейвлетів найчастіше використовується базис, заснований на добре локалізованому у часовій та в частотній областях вейвлеті Морле. Характерний параметр ω_0 дозволяє змінювати вибірковість базису. Дійсна та уявна частини $\psi(t)$ – це амплітудно-модульовані коливання.

У наш час вибір вейвлетов досить великий. Тільки в пакеті Wavelet Toolbox (MATLAB) налічується півтора десятки материнських вейвлетів; при цьому ряд з них має ще безліч варіантів. Для одержання довідки про будь-який тип вейвлета при роботі в командному режимі MATLAB досить виконати команду waveinfo ('type'), вказавши тип вейвлета. Для перегляду вейвлетів досить виконати команду wavemenu, і у вікні, що з'явилося, зі списком розділів вейвлет-перетворень натиснути кнопку Wavelet Display. Натискання цієї кнопки виводить вікно перегляду вейвлетів, у якому можна переглядати обраний вейвлет (з ім'ям 'Name') та інформацію про нього.

Приклади материнських вейвлетів

Основні функції, які створюють вейвлети, так звані материнські вейвлети, наведені в табл. 2.1.

Таблиця 2.1 – Материнські вейвлети

Вейвлети	**Аналітичний запис** $\psi(t)$	**Спектральна щільність** $\psi(\omega)$
Дійсні безперервні базиси		
Гауссові: – першого порядку або WAVE-вейвлет – другого порядку або МНАТ-вейвлет «мексиканський капелюх» – n-порядку	$-t\exp(-t^2/2)$ $(1-t^2)\exp(-t^2/2)$ $(-1)^n\frac{d^n}{dt^n}[\exp(-t^2/2)]$	$(i\omega)\sqrt{2\pi}\exp(-\omega^2/2)$ $(i\omega)^2\sqrt{2\pi}\exp(-\omega^2/2)$ $(-1)^n(i\omega)^n\sqrt{2\pi}\exp(-\omega^2/2)$
DOG-difference of gaussians	$e^{-t^2}2-0{,}5e^{-t^2/8}$	$\sqrt{2\pi}(e^{-\omega^2/2}-e^{-2\omega^2})$
LP-Littlewood&Paley	$(\pi t)^{-1}(\sin 2\pi t-\sin \pi t)$	$\begin{cases}(2\pi)^{-1.2}, & \pi\le\lvert t\rvert\le 2\pi\\ 0, & \textit{в іншому випадку}\end{cases}$
Дійсні дискретні базиси		
HAAR-вейвлет	$\ge\begin{cases}1, & 0\le t\le 1/2\\ -1, & 1/2\le t\le 1\\ 0, & t<0,\ t>0\end{cases}$	$ie^{i\omega/2}\frac{\sin^2\omega/4}{\omega/4}$
FHAT-вейвлет, або «французький капелюх»	$\ge\begin{cases}1, & \lvert t\rvert\le 1/3\\ -1/2, & 1/3\le\lvert t\rvert\le 1\\ 0, & \lvert t\rvert>1\end{cases}$	$\frac{4}{3}\frac{\sin^3\omega/3}{\omega/3}$
Комплексні		
Морле (Morlet)	$e^{i\omega_0 t}e^{-t^2/2}$	$\sigma(\omega)\sqrt{2\pi}e^{-(\omega-\omega_0)^2/2}$
Пауля (Paul) – чим більше n, тим більше нульових моментів має вейвлет	$\tilde{A}(n+1)\frac{i^n}{(1-n)^{n+1}}$	$\sigma(\omega)\sqrt{2\pi}(\omega)^n e^{-\omega}$

Вище був наведений невеликий перелік типів вейвлетів, що аналітично описуються в явному вигляді. Однак більшість типів вейвлетів не має аналітичного опису у вигляді однієї формули, а задається ітераційними виразами, які легко обчислюються комп'ютерами. Прикладом таких

вейвлетів є функції Добеші (Daubechies), одна з яких (db4) використовується як вбудована в MathCad.

Вибір конкретного материнського вейвлета цілком залежить від характеру поставленого завдання та від конкретного аналізованого сигналу.

При обробці зображень доводиться мати справу з двовимірними масивами S(x, y). Вони задаються в просторі $V = \{x, y\} \in R^2$ як функції двох змінних x і y. У цьому випадку двовимірна вейвлет-функція має вигляд:

$$\frac{1}{\sqrt{a_1 a_2}} \psi\left(\frac{x-b_1}{a_1} \cdot \frac{x-b_2}{a_2}\right), \tag{2.44}$$

де a_1 і a_2, b_1 і b_2 – значення a і b для кожного виміру.

Для ВП дискретних зображень батьківський та материнський вейвлети будують таким чином:

$$\phi(x, y) = \varphi(x)\varphi(y) , \tag{2.45}$$

$$\begin{aligned} &\psi_{LH}(x, y) = \varphi(x)\psi(y), \ \psi_{HL}(x, y) = \psi(x)\varphi(y), \\ &\psi_{HH}(x, y) = \psi(x)\psi(y), \end{aligned} \tag{2.46}$$

де індекси H і L означають реалізацію фільтрів високих частот та низьких частот складових.

Тоді двовимірні вейвлети запишуться в такому вигляді:

$$\left.\begin{matrix} 2^{-m}\varphi(2^{-m}x-k)\varphi(2^m y-1), \ 2^{-m}\varphi(2^{-m}x-k)\psi(2^m y-1) \\ 2^{-m}\psi(2^{-m}x-k)\varphi(2^m y-1), \ 2^{-m}\psi(2^{-m}x-k)\psi(2^m y-1) \end{matrix}\right\} . \tag{2.47}$$

Таким чином, на двовимірній площині відбувається аналіз по горизонталі, вертикалі й діагоналі з однаковим розширенням відповідно до трьох наведених вище вейвлетів.

Формули дискретного ВП двовимірних сигналів і зображень, створені з урахуванням наведених вище співвідношень (2.47), використані в пакеті Wavelet Toolbox.

Пряме ВП зображення відбувається таким способом. Припустимо, що маємо зображення розміром $N \times N$ (рис. 2.16, а). Спочатку кожний з N рядків зображення розділяється (фільтрується) на низькочастотну (НЧ) і високочастотну (ВЧ) половини. У результаті виходить два зображення розміром $N \times N/2$ (рис. 2.16, б). Далі кожний стовпець ділиться так само, у підсумку виходить чотири зображення розміром $N/2 \times N/2$ (рис. 2.16,

в): НЧ по горизонталі та вертикалі (НЧНЧ1), ВЧ по горизонталі та вертикалі (ВЧВЧ1), НЧ по горизонталі та ВЧ по вертикалі (НЧВЧ1) і ВЧ по горизонталі та НЧ по вертикалі (ВЧНЧ1). Перше із зазначених вище зображень ділиться аналогічним чином на наступному кроці (рівні) перетворення (рис. 2.16, г) і т. д.

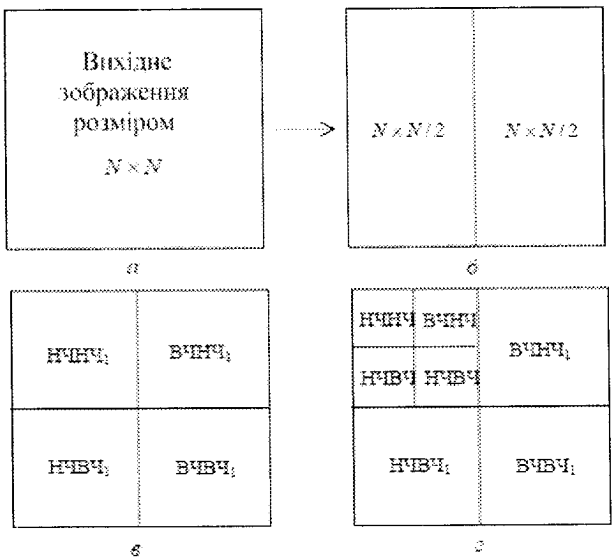


Рисунок 2.16 – Пряме ВП зображення

Для прикладу розглянемо як можна здійснити видалення шумів та компресію зображень за допомогою ВП. Для вирішення даної задачі здійснюється граничне обмеження рівня коефіцієнтів деталізації. Задавши певний поріг і «відтинаючи» коефіцієнти нижче цього порога, можна значно знизити рівень шуму і стиснути зображення.

Нижче наведений фрагмент програми фільтрації зображення від шумів, що завантажений з файлу noise. При цьому використані функції wpbmpen (встановлення глобального порога) і wpdencmp (видалення шумів і стиск зображень) :

```
load noissi2d; nbc = size(map,1); wname = 'db8';
lev = 2; tree = wpdec2(X,lev,wname);
det1 = [wpcoef(tree,2) wpcoef(tree,3) wpcoef(tree,4)];
sigma = median(abs(det1(:)))/0.6745; alpha = 1.1;
71
thr = wpbmpen(tree,sigma,alpha); keepapp = 1;
xd = wpdencmp(tree,'s','nobest',thr,keepapp);
colormap(pink(nbc));
subplot(221), image(wcodemat(X,nbc));
```

```
title('Вихідне зображення')
subplot(222), image(wcodemat(xd,nbc));
title('Відфільтроване зображення')
end
```

На рисунку 2.17 наведені вихідне і відфільтроване зображення, отримані при виконанні програми.

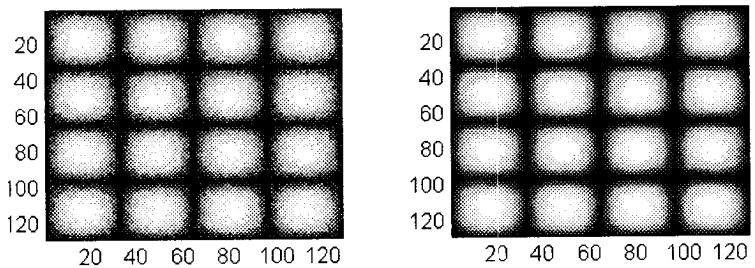


Рисунок 2.17 – Вихідне і відфільтроване зображення

Ряд прикладів з аналізу та реконструкції зображень, їх компресії та фільтрації від шуму наведений у розділі GUI Wavelet Toolbox, що активізується кнопкою Wavelet-2D.

На рис. 2.18 наведено приклад застосування Wavelet-2D: у верхньому лівому кутку наведено реальне зображення; у нижньому правому кутку дано його вейвлет-розклад (dwt) на прямокутні сегменти; у лівому нижньому кутку реконструкція зображення (idwt); у правому верхньому кутку ілюструється ділянка декомпозиції зображення.

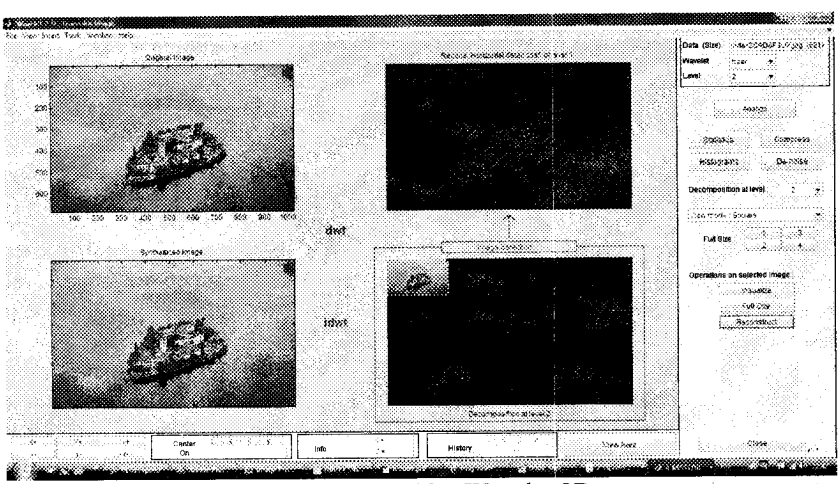


Рисунок 2.18 – Wavelet-2D

До стиску зображень проявляється значний інтерес в усьому світі. Це зумовлено стрімким розвитком цифрової техніки обробки зображень, кольорових принтерів, графічних моніторів, цифрових фото- і відеокамер тощо. Зображення, подане в цифровому вигляді, має досить великий обсяг у бітах. Наприклад, кольорове зображення розміром 512×512 вимагає для свого зберігання 768 кбайтів, а якщо передавати відеопослідовність таких зображень зі швидкістю 25 кадрів у секунду, то необхідна швидкість складе 188 Мбітів/с.

Принципова відмінність процедури компресії за допомогою ДВП від широко розповсюдженого стиску за стандартом JPEG полягає в тому, що він працює з усім зображенням, у той час як в JPEG зображення розбивається на блоки, які стискуються незалежно. У випадку ВП можна підібрати таку базисну вейвлет-функцію, що адаптована до найбільш інформативних особливостей зображення. Під адаптивністю розуміють, що елементи або ділянки зображення з досить плавною зміною яскравості подаються невеликим числом вейвлет-коефіцієнтів.

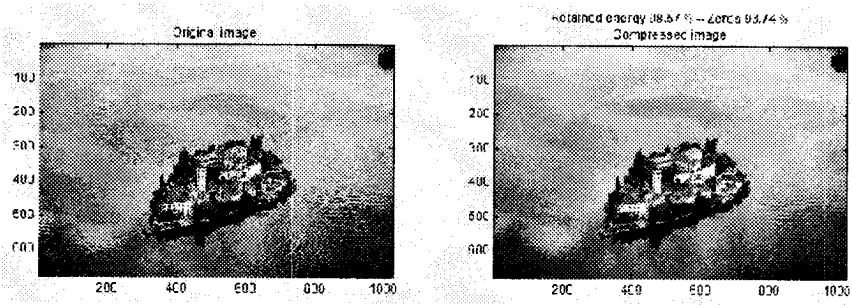


Рисунок 2.19 – Стиск зображення з використанням ВП

2.4 Нейронні мережі в задачах обробки зображень

Для розпізнавання складних об'єктів створюють системи на основі нейронних мереж (НМ, neural network). Вони можуть мати топологію, орієнтовану на розв'язання конкретної задачі з урахуванням властивостей об'єкта – просторово-часову орієнтацію, масштаб, геометричні параметри об'єкта, охоплюючи координати, кутове положення, лінійний розмір, відстань, тощо. У той же час істотним недоліком типових НМ є відсутність ефективних засобів для розв'язання задач розпізнавання динамічних образів. Основною проблемою інтерпретації динамічних візуальних сцен є висока розмірність простору ознак, наявність геометричних перетворень над об'єктом. Стиск простору ознак виконують методом знаходження інтегральних і інваріантних до геометричних

перетворень параметрів зображень. Метод геометричних та більш загальних алгебраїчних інваріантів відіграє значну роль у розв'язанні задач розпізнавання зображень. Так, наприклад, інваріанти, у тому числі інваріантні моменти, були успішно використані для розпізнавання профілів літаків і танків, друкованих і рукописних букв, параметрів стикувального вузла космічного апарата, а також багатьох інших об'єктів. Математичне обґрунтування інваріантних особливостей напівтонових зображень базується на теорії алгебраїчних інваріантів.

Суть НМ полягає в тому, що мережа складається з елементів, котрі називаються формальними нейронами (formal neuron). Кожен нейрон приймає набір сигналів, що надходять на його входи від одної групи таких же нейронів, обробляє сигнали з врахуванням попередніх сигналів і адаптації до них на основі процедур навчання та передає результати обробки другій групі нейронів. Зв'язки між нейронами кодуються вагами, що відображають важливість їх інформації для визначення загального результату. Основний принцип настроювання нейронної мережі полягає в застосуванні процедур оптимізації та адаптації на основі певних критеріїв, здатності до перенавчання. Однією з переваг НМ є те, що всі елементи можуть функціонувати паралельно, тим самим істотно підвищуючи ефективність розв'язання задач, особливо при обробці зображень в реальному часі. Системи розпізнавання об'єктів зображення, що засновані на нейронних мережах, використовують ієрархічну архітектуру. Спочатку вектор ознак обробляється грубою, з високим рівнем похибок, але швидкою мережею, далі, якщо вектор не був класифікований як необ'єкт, алгоритм розв'язання коректується більш точною і більш повільною мережею.

Переважна кількість прикладних нейронних систем передбачає використання багатошарових персептронів (назва «персептрон» походить з англійського perceptron – сприйняття, оскільки перші зразки таких структур призначались для моделювання зору). Популярність персептронів зумовлена широким колом доступних для них задач. Загалом вони вирішують задачу апроксимації багатовимірних функцій, іншими словами – побудову багатовимірного відображення $F : x \rightarrow y$, яке узагальнює заданий набір прикладів (еталонних пар даних) $\{x^a, y^a\}$.

Залежно від типу вихідних змінних (тип вхідних не має вирішального значення) апроксимація функції може набувати вигляду:

- класифікації (дискретний набір вихідних значень);
- регресії (неперервні вихідні дані).

Множина практичних задач розпізнавання зображень, фільтрації шумів, передбачення часових рядів та інших зводиться до цих базових задач.

Розглянемо алгоритм навчання персептрона на простій модельній задачі. Персептрон навчають, подаючи сукупність (множину) зображень

по одному на його вхід і змінюють ваги доти, доки для всіх зображень не буде досягнуто необхідний вихід. Припустимо, що вхідні зображення нанесено на демонстраційні карти. Кожну карту розбито на квадрати і від кожного квадрата на персептрон подається вхідний сигнал. Якщо в квадраті є лінія, то від неї подається одиниця, у протилежному випадку – нуль. Сукупність квадратів на карті задає сукупність нулів і одиниць, котрі подаються на входи персептрона. Мета полягає в тому, щоб навчити персептрон вмикати індикатор за умови подавання на нього сукупності входів, що задають непарне число, і не вмикати у випадку парного. На рисунку 2.20 показана така персептронна конфігурація.

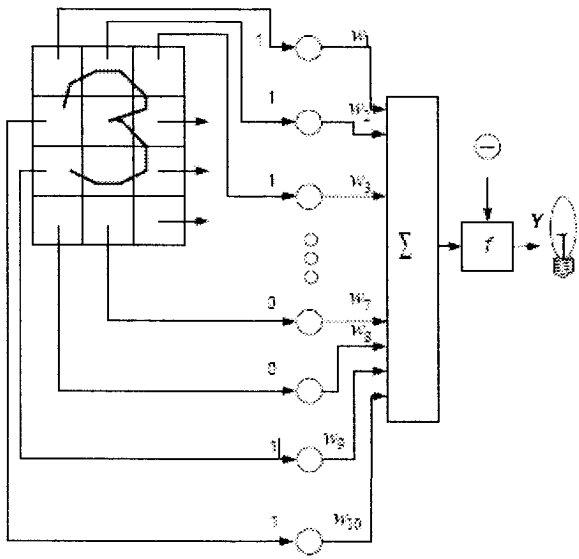


Рисунок 2.20 – Персептронна система розпізнавання зображень

Припустимо, що вектор x є зображенням демонстраційної карти, яка піддається розпізнаванню. Кожну компоненту x_i (квадратик зображення карти) вектора **x** перемножують на відповідну компоненту w_i вектора ваг w. Ці добутки підсумовують. Якщо сума перевищує поріг q , то вихід нейрона y дорівнює одиниці (індикатор засвічується), у протилежному випадку – нуль. Цю операцію компактно записують у векторній формі: $y = xw$. Для навчання мережі образ x подають на вхід і обчислюють вихід y. Якщо вихід правильний, то нічого не змінюється. Однак якщо вихід неправильний, то ваги, приєднані до входів, що підсилюють помилковий результат, модифікуються, щоб зменшити помилку.

Пересвідчимось, як це відбувається. Припустимо, що демонстраційна карта з цифрою 3, подана на вхід і вихід y, дорівнює одиниці (мережа

вказує на непарність). Оскільки це правильна відповідь, то ваги не змінюються. Якщо на вхід подають карту з номером 4 і вихід у дорівнює одиниці (непарне число), то ваги, приєднані до одиничних входів, повинні бути зменшені, оскільки вони прагнуть дати неправильний результат. Аналогічно, якщо карта з номером 3 дає нульовий вихід, то ваги, приєднані до одиничних входів, необхідно збільшити, щоб скорегувати помилку.

За скінченне число кроків мережа навчиться розділяти карти на парні і непарні за умови, що сукупність цифр лінійно роздільна. Отже для всіх непарних карт вихід буде більшим від порогу, а для всіх парних – нижчим. Зазначимо, що це навчання глобальне, тобто мережа навчається на всій можливій множині вхідних сигналів.

Причина популярності персептронів в тому, що для свого кола задач вони є універсальними та ефективними, з погляду обчислювальної складності, пристроями.

Недоліком використання нейронних мереж є їх перенавантаження при надмірному збільшенні кількості нейроні у мережі. Іншим недоліком є те, що існує великий клас функцій, які неможливо розділити за допомогою одношарової мережі. Про ці функції говорять, що вони є лінійно нероздільними, і саме вони накладають вагомі обмеження на можливості одношарових мереж.

Контрольні запитання та завдання

1. У чому полягає сутність поелементної обробки зображень?
2. Доведіть тотожність прямого й оберненого двовимірних ДПФ.
3. Поясніть, чому при обмеженому розмірі околу, що застосовується при КІХ-фільтрації, не можна досягти граничного заглушення шуму?
4. Назвіть умови, при виконанні яких інверсна фільтрація забезпечує високу якість відновлення зображень.
5. Яка структура двовимірного частотного спектра дискретного зображення?
6. За яких умов, використовуючи дискретне зображення, можна без втрат відновити безперервне?
7. Доведіть, що двовимірний фільтр із прямокутною частотною характеристикою ідеально відновлює безперервне зображення з дискретного.
8. Які методи покращення якості зображення ви знаєте?
9. Реалізуйте в ППП Matlab метод покращення якості зображення шляхом вирівнювання гістограми яскравості пікселів за допомогою функції *Image Processing Toolbox – histeq*.

10. За допомогою яких функцій можна накласти сторонній шум на зображення? Які види шуму ви знаєте?
11. В чому полягає суть двовимірної згортки зображення?
12. Розгляньте використання функції *roifill* для усунення дефектів напівтонового зображення.
13. Реалізуйте за допомогою функції *nlfilter* операцію усереднення з порогом в цілях фільтрації імпульсного шуму.
14. Перевірте чи є зображення напівтоновим, бінарним, палітровим чи повнокольоровим за допомогою функцій *isind, isgray, isrgb,isbw.*
15. Реалізуйте перетворення повнокольорового зображення в напівтонове, а напівтонового в палітрове за допомогою функцій *im2double, gray2ind.*
16. Порівняйте алгоритми побудови множини Мандельброта та сніжинки Коха.
17. Який показник використовується для зв'язку фрактальної та топологічної розмірностей?
18. Порівняйте алгоритми стиску зображень (RLE, Хаффмана, LZW). В чому полягає особливість фрактального стиску зображень?
19. Поясніть поняття: розмірність Хаусдорфа-Безиковича деякої множини A.
20. Які перетворення називаються масштабними? Міра Хаусдорфа має властивість інваріантності щодо масштабних перетворень?
21. Сформулюйте загальні вимоги, які повинна задовольняти розмірність множини при будь-якому способі виміру цієї множини.
22. На основі яких функцій будуються найбільш поширені материнські базиси? Чим це обумовлено?
23. Яким чином виконується пряме вейвлет-перетворення?
24. Порівняйте процедури компресії зображення за допомогою ДВП та фрактальних методів.
25. Чим відрізняються нейронні мережі від інших типів мереж?
26. Яким чином відбувається навчання нейронної мережі?

РОЗДІЛ 3 ІНТЕРВАЛЬНИЙ АНАЛІЗ

Основний інструмент, що використовується в інтервальному аналізі (interval analysis), оснований на дуже простій ідеї оточення дійсних чисел інтервалами, а векторів областями прямокутної форми – паралелетопами. При цьому вперше з'являється можливість отримати гарантовану оцінку результатів комп'ютерних обчислень прямим переходом до інтервальних змінних в класичних чисельних алгоритмах, що використовуються зазвичай в обчисленнях з плаваючою точкою. Зовсім нещодавно інтервальний аналіз дозволив будувати алгоритми диференціювання, спеціально розроблені для роботи з множинами, що не мали раніш аналогів. Це дало можливість використовувати чисельні методи для доведення тверджень стосовно множин. Таким чином, алгоритми, що базуються на інтервальному аналізі, доповнюють алгоритми на основі комп'ютерної алгебри. Алгоритми на основі інтервального аналізу мають ту перевагу, що можуть працювати з більш загальним класом задач і з задачами, що можуть бути вирішені лише чисельно (наприклад, знаходження коренів поліноміального рівняння високого степеня), але при цьому розв'язані гарантовано. Відмітимо, що звичайні чисельні методи зі статичним пошуком (Монте-Карло) або підлаштуванням сітки не можуть бути використані для доведення навіть таких простих властивостей, як пустота і неоднозв'язність множини.

Окрім цього інтервальні методи знайшли широке застосування в моделюванні різноманітних систем, робастному аналізі.

3.1 Класична інтервальна арифметика

Інтервальну арифметику можна визначити так:

$$[a,b]*[c,d]=\{x*y \mid a\le x\le b,\ c\le y\le d\}, \tag{3.1}$$

де $*\in\{+,-,\cdot,/\}$. При цьому, якщо $*$ означає ділення, то $0\in[c,d]$. Неважко показати, що ці операції в кожному конкретному випадку еквівалентні нижченаведеним:

$$\begin{aligned}
&[a,b]+[c,d]=[a+c,b+d],\\
&[a,b]-[c,d]=[a-d,b-c],\\
&[a,b]\cdot[c,d]=[\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)],\\
&[a,b]/[c,d]=[a,b]\cdot[1/d,1/c],\ 0\in[c,d].
\end{aligned} \tag{3.2}$$

Окрім того, з формули (3.2) слідує, що:

1) віднімання необернене додаванню;

2) ділення необернене множенню;

3) інтервальне додавання і інтервальне множення асоціативні і комутативні;

4) закон дистрибутивності не виконується, а виконується включення $A(B+C) \subseteq AB+AC$, що називається властивістю субдистрибутивності.

Додавання і множення мають звичайні властивостями асоціативності та комутативності:

$$A+(B+C)=(A+B)+C,\ A\cdot(B\cdot C)=(A\cdot B)\cdot C,$$
$$A+B=B+A,\ A\cdot B=B\cdot A.$$

Нулем додавання є число 0, а одиницею множення – число 1:

$$0+A=A+0=A,\quad 1\cdot A=A\cdot 1=A.$$

Однією з властивостей інтервальної арифметики є мотонність за включенням. Це означає, що з $A \subseteq C$ і $B \subseteq D$ випливає $A * B \subseteq C * D$ при $* \in \{+,-,\cdot,/\}$.

Згідно з визначенням інтервальної арифметики та відповідно до символіки, що є загальною для робіт з інтервального аналізу, наведемо певні додаткові позначення:

a = $[\underline{a}; \overline{a}]$ – певний інтервал з множини **IR,** нижньою границею якого є $\underline{a}$, а верхньою – $\overline{a}$, тобто $\underline{a} \le \overline{a}$, і $\underline{a}, \overline{a} \in R$;

$|\boldsymbol{a}| = \overline{a} - \underline{a}$ – довжина інтервалу, легко бачити, що $|\boldsymbol{a}| \ge 0$.

Далі ми будемо позначати через маленьку непотовщену літеру x – дійсний аргумент дійсної функції, $x \in$ R, а через велику літеру **X** – інтервальний аргумент інтервальної функції, $\mathbf{X} \in \mathbf{IR}$;

Будь-яке дійсне число b може бути подане у вигляді інтервалу $[\underline{a}; \overline{a}]$, для якого справедливі співвідношення:

$$b=\overline{a}=\underline{a},$$
$$|a| =\overline{a}-\underline{a}=0.$$

Крім наведених базових арифметичних операцій можна навести правило розрахунку значення інтервалів, що є результатом степеня:

$$A^K = \begin{cases} [a^k, b^k] \text{ при } k = 2j+1, \\ [a^k, b^k] \text{ при } k = 2j, a \geq 0, \\ [b^k, a^k] \text{ при } k = 2j, b \leq 0, \\ [0, \max(a^k, b^k)] \text{ при } k = 2j0 \in A. \end{cases} \quad (3.3)$$

та правило множення інтервалу на дійсну константу:

$$\alpha \cdot [a;b] = \begin{cases} [\alpha \cdot a; \alpha \cdot b], \ \alpha \geq 0 \\ [\alpha \cdot b; \alpha \cdot a], \ \alpha \leq 0 \end{cases}. \quad (3.4)$$

Однією з важливих та визначальних властивостей при виконанні операцій в інтервальній арифметиці є властивість вкладення. Формально ця властивість визначається так. Якщо інтервали подають будь-яке фіксоване дійсне число в своєму діапазоні, то і результат арифметичної операції є будь-яким можливим дійсним числом в діапазоні інтервалу. В символьному записі, ця властивість наводиться таким чином.

Одновимірна функція g^{IR} має властивість вкладення, якщо:

$$\forall[(i) \in IR], \ \forall[x \in i], g^R(x) \in g^{IR}(i). \quad (3.5)$$

Двовимірна функція g^{IR} має властивість вкладення, якщо:

$$\forall[(i,j) \in IR], \ \forall[(x,y) \in (i,j)], \ g^R(x,y) \in g^{IR}(i,j). \quad (3.6)$$

На основі основних алгебраїчних операцій та властивостей інтервального обчислення можна отримати набір елементарних функцій для інтервальних чисел.

3.2 Інтервальне розширення та звуження

Подання будь-якої загальної функції g^R у вигляді інтервальної функції g^{IR} є одною з найважливіших проблем інтервального аналізу.

Інтервальним розширенням функції $g^R(x)$, $x \in R$ назовемо такий елемент $g^{IR}(\mathbf{X})$, $\mathbf{X} \in \mathbf{IR}$, що при $x \in X$ виконується умова $g^R(x) \in g^{IR}(\mathbf{X})$.

Інтервальне розширення позначається як

$$\underset{x \to X}{Di} g^R(x) = g^{IR}(X). \quad (3.7)$$

З точки зору практичного застосування інтервальних методів буває доцільним визначати інтервальне розширення більш спеціально, а саме, задаючи засіб його отримання. Підтвердженням сказаного можуть служити нижченаведені приклади.

а) Функція $g^{IR}(\mathbf{X})$, що отримана заміною дійсного аргументу x в раціональній функції $g^{R}(x)$ інтервальним аргументом X з переходом в інтервальну арифметику, називається природним інтервальним розширенням. В цьому випадку справедлива така теорема.

Теорема 3.1. Якщо $g^{IR}(\mathbf{X})$ – природне інтервальне розширення $g^{R}(x)$ і кожна компонента вектора $X= (X_1,..., X_n)$ в $g^{IR}(\mathbf{X})$ зустрічається не більш одного разу, то

$$g^{IR}(\mathbf{X}) = \bigcup_{x \in X} g^{R}(x).$$

Як видно, в цьому випадку для знаходження області значень $g^{R}(x)$ при $x \in X$ достатньо обчислювати раціональний інтервальний вираз $g^{IR}(\mathbf{X})$, тоді як, оперуючи дійсними функціями, ми повинні визначити нескінченну множину $\{g^{R}(x) \mid x \in X\}$.

б) Якщо множину дійсних чисел R прирівняти до множини машинних чисел $R_м$, то тоді інтервальне розширення функції $g^{R}(x)$ буде мати вигляд

$$g^{IR}(\mathbf{X})=(g^{R}(x))_{\mathbf{M}}+[-\varepsilon(x), \varepsilon(x)],$$

де $(g^{R}(x))_{\mathbf{M}}$ – результат обчислення $g^{R}(x)$ на машині, а $\varepsilon(x)$ – абсолютна похибка $g^{R}(x)$, що дозволяє враховувати помилки машинного обчислення. При цьому $\varepsilon(x)$ оцінюється величиною $\varepsilon_0|(g^{R}(x))_{\mathbf{M}}|$, де ε_0 – мінімальне машинне число.

Шляхом оператора Di можна перейти до дій над інтервалами. Необхідність в подібному інтервальному розширенні може виникнути наприклад, при врахуванні помилок подання на ЕОМ констант, що входять в функцію $g^{R}(x)$.

Потрібно зауважити, що з $g^{R} = \varphi^{R}$ не слідує, що Di g^{R} = Di φ^{R}. Наведемо наочний приклад.

Нехай $g^{R}(x) = (\sqrt{x})^3 - 1$, $\varphi^{R}=((x/\sqrt{x}-1)(x+\sqrt{x}+1))$, $X_0=< 1, 4 >$. Для відповідних природних інтервальних розширень $g^{IR}(\mathbf{X}) = (\sqrt{X})^3 - 1$ та $\varphi^{IR}(\mathbf{X}) = ((X/\sqrt{X}-1)(X+\sqrt{X}+1))$ маємо $g^{IR}(\mathbf{X})=<0 , 7>$, $\varphi^{IR}(\mathbf{X}) = < -7/2, 21>$.

Оберненою операцією щодо інтервального розширення є інтервальне звуження.

Інтервальним звуженням функції F(X), $X \in \mathbf{IR}$, називають відображення f (x), $x \in \mathbf{IR}$, що отримується з F(X) при X=[x, x]=x. В символьному записі

$$\underset{X\to x}{Rs}\ F(X) = f(x). \qquad (3.8)$$

Потрібно звернути увагу на деякі аспекти. Наприклад, розглянемо звуження функції $g^{IR}(X)$ за X.

З класу еквівалентності $\underset{X\to x}{Rs}\ g^{IR}(X)$ завжди можна виділити відображення, інтервальне розширення яких охоплюють $g^{IR}(X)$. Для цього до $\underset{X\to x}{Rs}\ g^{IR}(X)$ достатньо додати деяке відображення φ(x), таке що:

$$\bigcup_{x\in X}\varphi^{R}(x) = 0, \text{ але } \underset{X\to x}{Di}\ \varphi^{R}(x) \neq 0 .$$

Наприклад, $\varphi^{R}(x)$=x-x. З іншого боку, розширення $g^{IR}(X)$ може бути таке, що з відповідного класу еквівалентності можна виділити відображення, інтервальні розширення яких можуть як вносити, так і вноситися в $g^{IR}(X)$.

Справді, нехай, наприклад, $g^{IR}(X)=X^2-5X$. Тоді $\underset{X\to x}{Rs}\ g^{IR}(X) = x^2 - 5x$. Розглянемо два елементи з класу еквівалентності $g^{R}(x) = x^2-5x$, а саме: $g_1^R(x) = x(x-5)$ та $g_2^R(x) = x^2 - 6x + x$. Для відповідних природних інтервальних розширень в точці X_0 = <2, 4> маємо $g_1^{IR}(X_0)$=<-12, -2>, $g_2^{IR}(X_0)$= <-18, 8>, в той час як $g^{IR}(X_0)$= <-16, 6>, тобто $g_1^{IR}(X_0) \subset g^{IR}(X_0) \subset g_2^{IR}(X_0)$.

Застосування інтервальних засобів припускає побудову інтервальних розширень, на які накладаються умови найбільшої звуженості. В теоремі 3.1 наведена достатня умова найбільшої звуженості для випадку раціональних функцій.

3.3 Диференціювання та інтегрування в інтервальному аналізі

Підхід до диференціювання та інтегрування функції в інтервальній математиці не суттєво, але відрізняється від звичайної математики. Проте останні дослідження вчених математиків наблизили ці процеси до того, що нам відомо з дійсної математики.

Один з перших способів розрахунку похідної від інтервальної функції базувався на так званих методах занурення. Але застосування цього способу на практиці виявилось не дуже зручним. Тому дослідники спрямовували свій пошук на більш практичні способи диференціювання. В наш час найбільш поширеним є підхід до диференціювання, який ввели болгарські вчені. Вони відповідно до дійсної математики дещо змінили підхід до визначення послідовності та визначили поняття збіжності

послідовності, знаходження границі послідовності в інтервальній математиці. Після визначення границі послідовності стало можливим сформулювати поняття диференціального обчислення для інтервальних функцій. Отримане визначення стало дуже схожим на те, яке ми звикли чути в математиці дійсних чисел.

Припустимо, що задана послідовність інтервалів $\{A_n\}$, $A_n \in IR$. Назвемо інтервал A, що є перетином всіх інтервалів, які охоплюють всі або майже всі, за винятком обмеженої кількості, інтервали послідовності $\{A_n\}$, а значення s – границею послідовності $\{A_n\}_1^\infty$:

$$s \lim_{n\to\infty} A_n = A. \tag{3.9}$$

Якщо $\{a_n\}_1^\infty$ – послідовність дійсних чисел, то $a = s \lim_{n\to\infty} a_n$ – це мінімальний інтервал, який охоплює всі граничні точки послідовності. Наприклад: $s \lim_{n\to\infty}(-1)^n = [-1,1]$. Відповідно до поняття границі послідовності вводиться поняття границі інтервальної функції.

Припустимо, що $F : \Lambda \to IR$, $\Lambda \subset R$, $x_0 \in \Lambda$, $f \in F$. Будемо називати інтервал A s-границею функції в точці x_0, якщо A є перетином всіх інтервалів, які містять в собі інтервали вигляду

$$s \lim_{n\to\infty} F(f, x_n), \tag{3.10}$$

де $x_n \in \Lambda$, $x_n \to x_0$.

Введемо визначення границі інтервальної функції G, заданої через дійсні граничні функції: $G(x) = [\underline{g}(x), \overline{g}(x)]$. Нехай G – інтервальна функція, визначена в околі точки $x_0 : \{x \mid 0 < |x - x_0| < \delta\}$. Назвемо s-границею функції G при $x \to x_0$ інтервал

$$s \lim_{x\to x_0} G(x) = \left[\underline{\lim_{x\to x_0}} \underline{g}(x), \overline{\lim_{x\to x_0}} \overline{g}(x) \right]. \tag{3.11}$$

Тоді можна ввести визначення похідної інтервальної функції:

$$G'(x) = s \lim_{h\to 0} \left(\frac{G(x+h) - G(x)}{h} \right). \tag{3.12}$$

Дефініція інтеграла інтервальної функції також має декілька різних формулювань. Але все ж найбільш правильним та легким для сприйняття є визначення за ідеями Мура.

Припустимо, що f – неперервна функція для якої існує інтервальне розширення F. Припустимо, що функція F є інтервальною функцією, визначеною для X ⊂ A, де A=[a,b], a<b, та звуження F є неперервна дійсна функція за A, F(x)=f(x), при x ∈ A та f(x)⊂F(x) при X⊂A. Функція

$$g(x) = \int_a^x f(t)dt, t \in [a,b],$$

має неперервну похідну g′(x)=f(x) та згідно з теоремою про середнє

$$g(x)=g(a)+f(a+\theta\cdot(x-a))(x-a),$$

для деякого θ ∈ [0,1]. Оскільки g(a)=0, відповідно

$$g(x) = \int_a^x f(t)dt = \int f(a+\theta\cdot(x-a))(x-a), \quad \theta \in [0,1].$$

Тоді

$$g(x) = \int_a^x f(t)dt \in F(a+[0,1]\cdot(x-a))(x-a),$$

а оскільки x∈ [a,b] і x≥a, то

$$\int_a^x f(t)dt \in F([a,x])(x-a).$$

Звідси для будь-якого X=$\underline{[x,\bar{x}]}\subset A$, позначивши $\int_{x_1}^{x_2} f(t)dt$ через $\int_{[x_1,x_2]} f(t)dt$ або $\int_X f(t)dt$, маємо

$$\int_X f(t)dt \in F(X)\omega(X). \qquad (3.13)$$

Використовуючи властивість адитивності інтеграла:

$$\int_{[x_1,x_2]} f(t)dt + \int_{[x_2,x_3]} f(t)dt = \int_{[x_1,x_3]} f(t)dt,$$

отримаємо

$$\int_{[x_1,x_3]} f(t)dt \in F([x_1,x_2])(x_2 - x_1) + F([x_2,x_3])(x_3 - x_2). \quad (3.14)$$

Формула (3.12) дає підстави для введення поняття інтервального інтеграла від інтервальної функції.

Якщо f – неперервна інтервальна функція дійсної змінної та F – неперервне інтервальне розширення f таке, що F(x) = f (x), то інтеграл визначається як

$$\int_{[a,x]} f(t)dt = \bigcap_{n=1}^{\infty} \sum_{i=1}^{n} F(X_i^{(n)}) \cdot \frac{x-a}{n}. \quad (3.15)$$

Якщо f – неперервна інтервальна функція дійсної змінної $x \in [a,b]$, то існує пара неперервних дійсних функцій f_1 і f_2 таких, що $f(x) = [f_1(x), f_2(x)]$, і введене визначення інтеграла рівносильне нижченаведеному:

$$\int_{[a,x]} f(t)dt = \left[\int_{[a,x]} f_1(t)dt, \int_{[a,x]} f_2(t)dt \right]. \quad (3.16)$$

3.4 Інтервальні методи розв'язання диференціальних рівнянь

Цікавість до використання інтервальних методів при розв'язанні диференціальних рівнянь пояснюється тим, що в рамках інтервального аналізу вхідні дані диференціального рівняння можуть бути задані у вигляді інтервалів, а отримані рішення враховують не лише помилки у вхідних даних, але й помилки апроксимації та округлення.

При інтервальному моделюванні однією з задач є розв'язання диференціальних рівнянь з інтервальними даними. На сьогодняшній день розроблено багато інтервальних методів розв'язання диференціальних рівнянь. Розглянемо детально найвідоміші методи.

3.4.1 Інтервальний метод другого порядку для розв'язання звичайних диференціальних рівнянь

Розглянемо задачу Коші

$$\frac{dy}{dx} = f(y), \quad y = y(x), \quad x \in R, \tag{3.17}$$

$$y(0) = y_0. \tag{3.18}$$

Припустимо, що функція $f(y)$ визначена і має дві перші обмежені похідні на інтервалі A=[a,b].

Розглядуваний нижче метод розв'язання задачі (3.17), (3.18) припускає неточно задані початкові дані, а саме: припустимо, що існує інтервал Y_0 такий, що він лежить строго в A і $y_0 \in Y_0$. Окрім того припустимо, що функція f(y) має інтервальне розширення F(Y), що має такі властивості:

1) функція F(Y) визначена і неперервна при всіх $Y \subset A$;
2) функція F(Y) монотонна за включенням, тобто з того, що $Y_1 \subset Y_2$, слідує $F(Y_1) \subset F(Y_2)$;
3) існує число $l>0$ таке, що $\omega(F(Y)) \le l\omega(Y)$ для всіх $Y \subset A$, а також існує $\Psi(Y)$ – інтервальне розширення функції $f(ff'' + (f')^2)$, що визначене при $Y \subset A$ і монотонне за включенням.

Оскільки Y_0 лежить в A, для деякого скінченного $h_0>0$ знайдеться таке число $\xi>0$, що

$$Y_0 + \xi(F(A) - h_0^2\Psi(A)/12) \subset A.$$

Інтервальний розв'язок побудуємо на відрізку $[0, \xi]$, для цього розіб'ємо його на m частин точками $x_i = ih\,(i=0,1,\ldots,m)$, $h = \xi/m < h_0$.

Теорема 3.2. Якщо інтервали $Y(x_i)=Y_i$ $(i=0,1,...,m)$ визначаються формулами

$$Y_0 = Y(x_0) = Y(0), \tag{3.19}$$

$$Y_{i+1} = Y_i + (h/2)\{F(Y_i) + F(Y_i + hF(Y_i + [0,h]\cdot F(A))) - (h^3/12)\Psi(Y_i + [0,h]\cdot F(A))\}, \tag{3.20}$$
$$(i=0,1,...,m),$$

то для будь-якого розв'язку $y(x)$ рівняння (3.17) такого, що $y(0) \in Y_0$, справедливі включення $y(x_i) \in Y_i (i=1,...,m)$ і має місце така оцінка для ширини інтервалів Y_i:

$$\omega(Y_i) \le Nh^2 + M\omega(Y_0), \tag{3.21}$$

де N і M – дійсні константи, що не залежать від i і h.

3.4.2 Інтервальні методи типу Рунге-Кутта

Розглянемо задачу Коші:

$$\frac{dy}{dx} = f(x, y), \tag{3.22}$$

$$y(0) = y_0 \in Y_0. \tag{3.23}$$

Припустимо, що функція *f(x,y)* визначена для всіх $(x, y) \in \Delta_x \times \Delta_y$, де $\Delta_x = \{x \mid 0 \le x \le c\}$, $\Delta_y = \{y \mid a \le y \le b\}$.

Оскільки ряд параметрів, що визначаються формулами методу Рунге-Кутта, використовується при побудові відповідних інтервальних формул, розглянемо спосіб отримання розв'язку цим методом в класичній математиці.

Для знаходження *y(x+h)*, якщо відомо *y(x)*, використовується формула:

$$y(x + h) = y(x) + \sum_{i=1}^{q} p_i k_i(h), \tag{3.24}$$

де $k_1(h) = hf(x, y)$, $k_2(h) = hf(x + \alpha_2 h, y + \beta_{21} k_1(h)),\ldots,$

$k_q(h) = hf(x + a_q h, y + \beta_{21} k_1(h) + \ldots + \beta_{qq-1} k_{q-1}(h)),$

а величини $\alpha_2, \ldots, \alpha_q, p_1, \ldots, p_q, \beta_{ij}$ $(0<j<i\le q)$ залежать від вибору порядку похибки, *q* і самої функції *f*.

Нехай функція *f(x,y)* має інтервальне розширення ***F(X,Y)***, що має властивості:

1. ***F(X,Y)*** визначена і неперервна для всіх $X \subset \Delta_x$, $Y \subset \Delta_y$;

2. ***F(X,Y)*** монотонна за включенням, тобто з $X_1 \subset X$, $Y_1 \subset Y$ слідує, що $F(X_1,Y_1) \subset F(X,Y)$;

3. Існує константа L>0, така, що $\omega(F(X,Y)) \le L(\omega(X) + \omega(Y))$ для всіх $X \subset \Delta_x$, $Y \subset \Delta_y$, де $\omega([a,b]) = b - a$. Нехай, окрім цього, ψ(x,y) має інтервальне розширення $\Psi(X,Y)$, визначене для всіх $X \subset \Delta_x$, $Y \subset \Delta_y$ і монотонне за включенням.

Тоді розв'язок інтервального диференціального рівняння буде таким:

$$Y_m(x_{j+1}) = Y_m(x_j) + \sum_{i=1}^{q} p_i k_i^j(h) + (\Psi(X_j, Y_m(x_j)) + [-\alpha, \alpha])h^{s+1}. \tag{3.25}$$

Величини $k_i^j(h)$ та α обчислюються за формулами:

$$k_1(h) = hF(X, Y),$$
$$\mathrm{k}_2(h) = hF(X + \alpha_2 h, Y + \beta_{21} k_1(h)),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$k_q(h) = hF(X + a_q h, \mathrm{Y} + \beta_{21} k_1(h) + \ldots + \beta_{qq-1} k_{q-1}(h)), \qquad (3.26)$$
$$\alpha = Mh_0,$$
$$\left|\varphi^{(s+2)}(\theta h)/(s+2)!\right| \le M \le \infty,$$

де s – порядок рівняння.

Цей метод справедливий і для розв'язання рівнянь зі змінним кроком, а також для систем рівнянь вигляду (3.22) з відповідними початковими умовами.

3.4.3 Метод Крукеберга

Трикроковий метод Крукеберга чисельного розв'язання задачі Коші для звичайних диференціальних рівнянь дозволяє отримати інтервали, що містять розв'язок задачі. В ньому попередньо будується інтервальний розв'язок для фіксованої дійсної задачі з певнимим початковими значеннями з заданої множини (інтервалу), а потім методом збурень знаходяться інтервальні включення для всіх можливих рішень.

Нехай рівняння першого порядку $\frac{dy}{dx} = f(x, y)$ з початковими умовами $y(x_0)=y_0$ має єдиний розв'язок $\tilde{y}(x; x_0, y_0)$ на $[x_0,x_1]$. Якщо початкові дані задані неточно, тобто $y_0 \in Y_0$, де Y_0 – певний інтервал в R, то розв'язки задачі в такій постановці утворюють множину $\overline{Y} = \{z \mid z = \tilde{y}(x; x_0, y_0), y_0 \in Y_0\}$. Необхідно знайти інтервал $Y_1^* \supseteq Y_1$, де $Y_1 = \overline{Y}(x_1)$, $x_1=x_0+h$, величина кроку h визначається в процесі розв'язання.

На першому кроці знаходимо крок h і інтервальний поліном k-го степеня $P_k(x-x_0)$ такий, що при $x_1=x_0+h$, $P_k(x - x_0) \supseteq \overline{Y}(x)$, $x \in [x_0, x_1]$. Такі поліноми можуть бути отримані за ітераційним алгоритмом Пікара:

$$P_{k+1} = Y_0 + \int_0^x F(x_0 + \xi) d\xi, \quad (k = 0,1,2,\ldots), \qquad (3.27)$$

де F – інтервальне розширення функції f. На практиці обмежуються поліномами нульового степеня p_0.

На другому етапі знаходиться інтервальний розв'язок початкового

рівняння з дійсними початковими даними $y(x_0)=d_0 \in Y_0$. Використовуючи формулу Тейлора, в якій дійсні аргументи замінені інтервальнимим, маємо інтервал

$$D_1 = d_0 + hF(x_0, d_0) + \frac{h^2}{2!} F'([x_0, x_1], P_0([0, h])).$$

Зрозуміло, що $d_1 = \tilde{y}(x_1; x_0, d_0) \in D_1$.

Третій етап є інтервальним варіантом методу збурень. Інтервал $U_0=Y_0-d_0$ всіх можливих збурень початкового значення d_0. Перепишемо початкову задачу у вигляді

$$\begin{aligned} &d(u+d)/dx = f(x; u+d), \\ &u(x_0) = u_0 \in U_0 \end{aligned} \qquad (3.28)$$

Нам відомо, що $u(x_1) = d_1 = \tilde{y}(x_1; x_0, d_0)$, необхідно визначити $u_1 = \tilde{y}(x_1; x_0, u_0)$. Шляхом розкладання правої частини за формулою Тейлора в околі розв'язку $\tilde{y}(x_1; x_0, d_0) = d_1$ і, отримуючи члени першого порядку відносно h, маємо рівняння

$$\frac{du}{dx} = u\frac{\partial f}{\partial y} \qquad (3.29)$$

з початковими умовами (3.28). Тоді інтервал $U_1=QU_0$, де $Q = \sum_{k=1}^{\infty} \frac{h^k}{k!} F^k$, F – інтервальне розширення функції f_y' і є шуканим розв'язком задачі (3.29), (3.28).

Метод Крукеберга природним чином узагальнюється на випадок системи рівнянь. При цьому $\frac{\partial f}{\partial y}$ буде матрицею, елементи якої є інтервальними розширеннями функцій $\frac{\partial f_i}{\partial y_i}$.

3.5. Подання інтервальної функції через граничні дійсні функції

Подання інтервальної функції через дві граничні дійсні функції є одним з важливих підрозділів інтервального аналізу. Подібне подання дозволяє замість вивчення інтервальної функції зосередити увагу на двох дійсних функціях. Це не тільки в певних випадках спрощує розрахунки,

але й дає змогу уявити як поводить себе інтервальна функція в тих або інших випадках, отримати більш точні результати розрахунків.

Проблема подання інтервальної функції двома граничними дійсними полягає в знаходженні подання

$$F(X)=[f_1(t), f_2(t)], \tag{3.30}$$

де $f_1(t), f_2(t)$ – певні дійсні функції, які відповідають визначенню.

Наприклад, якщо на відрізку $0 \le x \le 3$ розглянемо функцію

$$F(x)=x^2-[2,4]\cdot x+[3,5], \tag{3.31}$$

тоді $f_1(t)=x^2-4\cdot x+3$, $f_2(t)=x^2-2\cdot x+5$.

Дійсно, кожному значенню аргументу з інтервалу $x \in [0,3]$ функція (3.31) ставить у відповідність певний інтервал [a,b]. Отже, множина значень функції на площині (x,y) являє собою область, обмежену кривими $y=x^2-4\cdot x+3$, $y=x^2-2\cdot x+5$ та відрізками прямих $x=0$ та $x=3$.

Процес визначення функцій f_1 та f_2 залежить від того, з яких причин функція F має за своє значення інтервали, тобто, чому результат є інтервальним числом. Таких причин можна виділити декілька:

1) функція F є функцією дійсного аргументу x, але константи, що входять до F, це інтервали;

2) функція F є функцією інтервального аргументу X, але константи, що входять до F, дійсні числа;

3) аргумент функції F та константи, що входять до її складу, є інтервали.

Якщо функція F є поліномом виду $F(x)=\sum_{i=1}^{m}[a_i;c_i]\cdot x^i$, то граничні функції виражаються як:

$$f_1(x)=\begin{cases}\sum_{i=0}^{m} a_i \cdot x^i & \text{якщо } x \ge 0,\\ \sum_{i=0}^{m}(a_i\cdot\theta(i)+c_i\cdot\gamma(i))\cdot x^i & \text{якщо } x \le 0,\end{cases} \tag{3.32}$$

$$f_2(x)=\begin{cases}\sum_{i=0}^{m} c_i \cdot x^i & \text{якщо } x \ge 0,\\ \sum_{i=0}^{m}(a_i\cdot\gamma(i)+c_i\cdot\theta(i))\cdot i & \text{якщо } x \le 0,\end{cases} \tag{3.33}$$

де

$$\gamma(i)=\begin{cases}0 & \text{якщо } i=2k,\\ 1 & \text{якщо } i=2k+1.\end{cases}$$

$$\theta(i)=\begin{cases}1 & \text{якщо } i=2k,\\ 0 & \text{якщо } i=2k+1.\end{cases}$$

Якщо повернутися до вищенаведеного прикладу, то отримаємо:

$$f_1(x)=\begin{cases}x^2-4x+3 & \textit{якщо } x\ge 0\\ x^2-2x+3 & \textit{якщо } x\le 0\end{cases},$$

$$f_2(x)=\begin{cases}x^2-2x+5 & \textit{якщо } x\ge 0\\ x^2-4x+5 & \textit{якщо } x\le 0\end{cases}.$$

Із застосуванням визначення граничних функцій дещо змінюється означення диференціала та інтеграла інтервальної функції.

Так, якщо f – неперервна інтервальна функція дійсного аргументу $x\in[a,b]$, то існує пара неперервних дійсних функцій f_1 та f_2 таких, що інтеграл функції можна визначити, як

$$\int\limits_{[a,x]} f(t)dt=\left[\int\limits_{[a,x]} f_1(t)dt;\ \int\limits_{[a,x]} f_2(t)dt\right]. \qquad (3.34)$$

Також можна змінити визначення границі інтервальної функції. Припустимо, що інтервальна функція G задана через дійсні граничні функції: $G(x) = [\underline{g}(x);\ \overline{g}(x)]$. Похідна для інтервальної функції G(x), що визначена в околі точки $x_0:\{x\mid 0<|x-x_0|<\delta\}$, може визначатись за формулою:

$$s\lim_{x\to x_0} G(x)=\left[s\lim_{x\to x_0}\underline{g}(x);s\lim_{x\to x_0}\overline{g}(x)\right]. \qquad (3.35)$$

Визначення інтервальної функції через її граничні широко використовується на практиці. Це насамперед пов'язано з тим, що апарат обчислень з дійсними функціями добре розроблений. Припустимо, що ми маємо інтервальний алгоритм, який необхідно реалізувати. Отже, очікувані результати – інтервали. Але з причин округлень, неможливості отримання

дійсної області значень інтервальної функції, а також не можливе отримання інтервалів, які значно ширші, ніж насправді.

Тому дуже часто йдуть іншим шляхом. Інтервальний алгоритм поділяють на два дійсних. Тоді кожний з дійсних реалізують в рамках інтервальної арифметики. Це не тільки зменшує складність розрахунків для людини, що не досить знайома з інтервальним аналізом, але в певних випадках дозволяє отримати більш точні результати. Але це можливо лише в випадках, коли подання інтервальної функції через її граничні можливе і не дуже складне, не займає зайвої кількості часу.

3.6 Розширення інтервальної арифметики

Класична інтервальна арифметика ***IR*** є неповною за своїм математичним змістом. З математичної точки зору, вона є лише комутативною напівгрупою за додаванням та множенням, а відносно порядку включення вона не є решіткою. Ця неповнота алгебраїчної та порядкової структур ***IR*** природно стимулює намагання створити на її основі більш досконалу інтервальну арифметику. І таке доповнення було зроблено в працях Каухера, а потім вдосконалено в працях Гарденеса та Трепата. Арифметика, що створена в цих наукових працях, отримала назву «розширеної інтервальної арифметики» чи «інтервальної арифметики Каухера».

Елементами інтервальної арифметики Каухера, як і в класичній інтервальній арифметиці, є дійсні пари $[\underline{x};\overline{x}]$, але вони не обов'язково зв'язані співвідношенням $\underline{x} \le \overline{x}$. Таким чином, множини інтервалів ***IR*** отримується шляхом приєднання невласних інтервалів $[\underline{x};\overline{x}]$, $\underline{x} > \overline{x}$ до множини $IR = \{[\underline{x};\overline{x}] \mid \underline{x},\overline{x} \in R, \underline{x} \le \overline{x}\}$ власних інтервалів та дійсних чисел. Власні та невласні інтервали, дві половинки IR, можна змінювати при відображенні дуалізації

$$dual : IR \to IR, \tag{3.36}$$

такому, що $dual[\underline{x};\overline{x}] = [\underline{x};\overline{x}]$.

Додавання та множення на дійсні константи визначається в арифметиці Каухера так:

$$[\underline{x};\overline{x}] + [\underline{y};\overline{y}] = [\underline{x} + \underline{y};\overline{x} + \overline{y}], \tag{3.37}$$

$$\lambda \cdot [\underline{x};\overline{x}] = \begin{cases} [\lambda \cdot \underline{x};\lambda \cdot \overline{x}], \text{якщо } \lambda \ge 0, \\ [\lambda \cdot \overline{x};\lambda \cdot \underline{x}], \text{якщо } \lambda \le 0. \end{cases} \tag{3.38}$$

Таким чином, кожен елемент x з IR має один протилежний елемент, що часто визначається як «opp **x**», і

$$opp[\underline{x};\overline{x}] = [-\underline{x};-\overline{x}]. \tag{3.39}$$

Часто в наукових працях можна зустріти спеціальне позначення для операції, що є оберненою до додавання, через Θ:

$$x \Theta y = x + opp\ y.$$

Віднімання та ділення в арифметиці Каухера визначаються як:

$$x\text{-}y = x + (-1)\text{-}y,$$
$$x / y = x \cdot \left[\frac{1}{\overline{y}};\frac{1}{\underline{y}}\right] \text{ при умові, що } 0 \notin \mathbf{y}.$$

Крім того, в інтервальній арифметиці Каухера зберігається монотонність інтервальних операцій за включенням:

$$\mathbf{x} \subseteq \mathbf{x}',\ \mathbf{y} \subseteq \mathbf{y}' \Rightarrow \mathbf{x} * \mathbf{y} \subseteq \mathbf{x}' * \mathbf{y}', \tag{3.40}$$

де $* \in \{+,-,\cdot,/\}$,x, x', y, y' $\in$ IR.

Дії над векторами та матрицями в розширеній інтервальній арифметиці Каухера визначають подібно тому, як це робиться в класичній арифметиці. Сума (різниця) двох інтервальних матриць однакового розміру є інтервальною матрицею того ж самого розміру, яка утворюється з поелементних сум (різниць) операндів. Якщо $A=(a_{ij}) \in IR^{m\times l}$, $A=(b_{ij}) \in IR^{l\times n}$, то результатом їх множення є матриця $C=(c_{ij}) \in IR^{m\times n}$ така, що

$$c_{ij} = \sum_{k=1}^{l} a_{ik} \cdot b_{kj}\ .$$

Багато означень та понять класичної інтервальної арифметики без змін переносяться в арифметику Каухера. Тому фактично Каухер не змінив основ, а тільки вніс певні вдосконалення. Деякі з них суттєво спростили застосування інтервального обчислення на практиці.

Контрольні завдання та запитання

1. Навести практичні приклади використання методів інтервального аналізу.
2. Сформулювати основні правила операцій над інтервалами.
3. Скласти алгоритми та програми реалізації найпростіших математичних операцій над інтервалами.
4. Подати у інтервальному вигляді такі функції: *sin, cos, tan, ctg, exp, ln.*
5. Скласти алгоритми та програми інтервальної реалізації математичних функцій, наведених у завданні 4.
6. Сформулюйте задачу Коші для інтервальних чисел. В чому відмінність інтервальних методів розв'язання диференціальних рівнянь від класичних?
7. Чим відрізняється класична інтервальна арифметика від інтервальної арифметики Каухера?
8. Побудувати інтервальне розширення для функцій:

$$f(x) = 2x^2 + 3x - 5;$$
$$f(x_1, x_2) = x_1 x_2 + x_1 + x_2 + 1.$$

9. Чи буде нижченаведена функція інтервальним розширенням деякої функції $f(x)$? Якщо так, то якої?

$$\text{а) } F(x) = 2[\underline{x}, \overline{x}] + [-1,1];$$

$$\text{б) } F(x) = \begin{cases} [0, \max\{\underline{x}^2, \overline{x}^2\}] + 2, & \underline{x}\overline{x} < 0; \\ [\underline{x}^2, \overline{x}^2] + 2, & \underline{x} \geq 0; \\ [\overline{x}^2, \underline{x}^2] + 2, & \underline{x} \leq 0. \end{cases}$$

10. Знайдіть інтервальний інтеграл на [a,b] для функції

$$F(x) = [1,\ 2]x^2 + [-1,\ 1]x + [1,\ 0].$$

11. Використовуючи формулу Тейлора знайдіть інтервальну оцінку інтеграла

$$\int_0^{\frac{\pi}{2}} \sin(x)dx.$$

12.Побудуйте інтервальний кубічний сплайн на відрізку [0,1] якщо $f_0 = [0.5, 1]$, $f_1 = [0,\ 0.5]$, $f_0' = [-0.5, -0.5]$, $f_1' = [-0.5, -0.5]$.

13.Знайти множину розв'язків системи лінійних алгебраїчних рівнянь

$$14.\ Ax = b,$$

15.де $A = \begin{pmatrix} [2,\ 3] & [-1,\ 1] \\ [-1,\ 1] & [2,\ 3] \end{pmatrix}$; $b = \begin{pmatrix} [0, 2] \\ [0, 2] \end{pmatrix}$.

14.Розв'яжіть систему рівнянь методом Гаусса

$$16.\ Ax = b,$$

17.де $A = \begin{pmatrix} [2,\ 3] & [-1,\ 1] \\ [-1,\ 1] & [2,\ 3] \end{pmatrix}$; $b = \begin{pmatrix} [0, 2] \\ [0, 2] \end{pmatrix}$.

РОЗДІЛ 4 МЕТОДИ ОПТИМІЗАЦІЇ І ПЛАНУВАННЯ

Методи оптимізації широко застосовуються для розв'язання задач теорії оптимальних процесів, оптимального регулювання, вироблення керувальних збурень на об'єкти. Без розробки та застосування методів оптимізації неможливе керування ректифікаційними колонами в спиртовій промисловості, установками крекінгу нафти, конверторами при виробництві сталі та ін. До транспортних задач та задачі комівояжера зводиться багато задач економічної кібернетики (мережне планування, управління запасами, перевезеннями та ін.), керування організацією виробництва (розподіл завдань, обробка деталей, конвеєрне виробництво) та задачі оптимального програмування. Окрема група задач теорії оптимізації – це задачі оптимального проектування. Наприклад, задачі проектування радіоелектронних засобів з заданими обмеженнями на рівень шуму та смугу пропускання чи показниками надійності в умовах старіння.

4.1 Класична постановка задачі оптимізації

Звичайна постановка задачі оптимізації така: в деякому просторі S тим чи іншим засобом виділяється деяка непуста множина M точок цього простору, яку називають припустимою множиною. Далі фіксується деяка дійсна функція $f(x)$, що задана в усіх точках x допустимої множини. Вона називається цільовою функцією. Задача оптимізації полягає в тому, щоб знайти точку x_0 в множині M, для якої функція $f(x)$ приймає екстремальне (максимальне або мінімальне) значення. В першому випадку для всіх точок x множини M задовольняється нерівність $f(x_0) \geq f(x)$, в другому випадку – нерівність $f(x_0) \leq f(x)$.

В практичних задачах можливі дві основні постановки оптимізаційних задач. В першому випадку задача розглядається в звичайному (евклідовому) просторі кінцевої розмірності. Точками x допустимої множини будуть кортежі $x = (x_1, x_2, ..., x_n)$ дійсних чисел, цільовою ж функцією $f(x) = F(x_1, ..., x_n)$ буде звичайна дійсна функція від n дійсних аргументів (n – розмірність простору). Таку задачу ми будемо називати в подальшому задачею оптимізації функцій. В другому випадку постановки оптимізаційної задачі як припустима множина виступає деяка множина M функцій дійсних змінних $y(x_1, ..., x_m)$, а цільовою функцією є деякий функціонал F, який ставить у відповідність кожній функції $y(x_1, ..., x_m)$ деяке дійсне число $F(y)$. Цю задачу ми будемо називати задачею оптимізації функціоналів або варіаційною задачею.

4.2 Класифікація задач оптимізації

Перш за все треба розділяти задачі параметричної та структурної оптимізації. Параметрична оптимізація є предметом, що розглядається в цьому розділі, де наведені постановка такої задачі та методи її розв'язання. Структурна оптимізація – це задача синтезу оптимальної структури системи, причому зміна структур та перетворення однієї структури в іншу здійснюється за спеціальним алгоритмом синтезу. Параметрична оптимізація об'єднує багато різних задач, що мають свої власні особливості та методи розв'язання. Класифікацію цих задач наведено на рисунку 4.1.

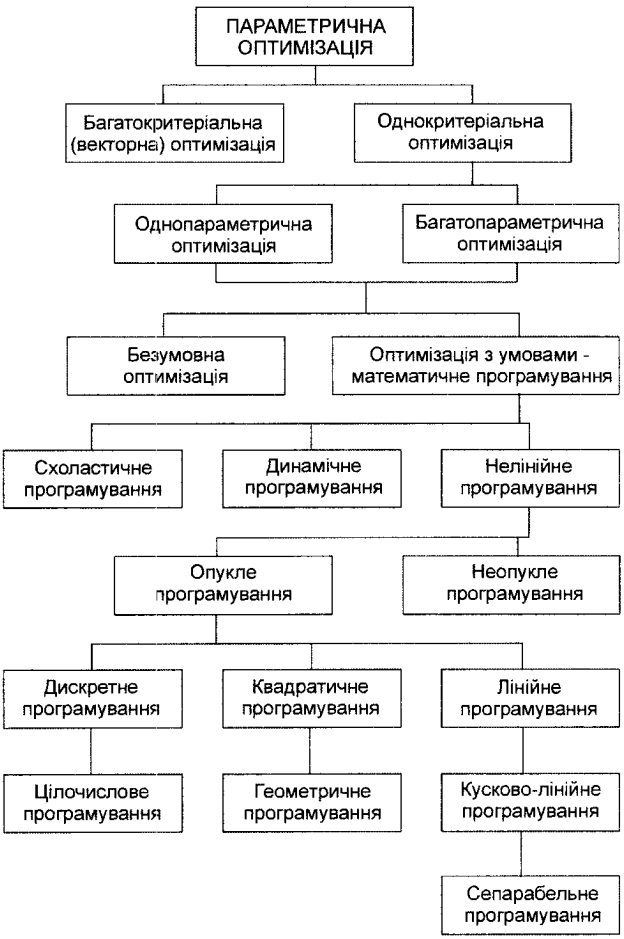


Рисунок 4.1 – Класифікація задач оптимізації

До цього треба додати деякий коментар:

1. Якщо існує декілька цільових функцій, то має місце задача векторної оптимізації.

2. Якщо кількість керованих параметрів X більше одиниці, то розв'язується задача багатопараметричної оптимізації.

3. Якщо існують обмеження та умови, що зв'язують параметри X, то виникає задача оптимізації з умовами, яка в кібернетиці дістала назву математичного програмування.

4. Математичне програмування об'єднує задачі нелінійного програмування (цільова функція в загальному випадку нелінійна), стохастичного програмування (параметри X – випадкова величина, а цільова функція – випадкова функція), динамічного програмування (оптимізація багатокрокових процесів пошуку рішення).

5. Якщо параметри, що керуються, приймають тільки дискретні значення, то виникає задача дискретної оптимізації, а якщо X – цілі числа, то – задача цілочислового програмування.

6. У випадку, коли цільова функція опукла та область, де задані X, теж опукла, то має місце задача опуклого програмування. Якщо цільова функція та умови лінійні – лінійного (кусково-лінійного) програмування; цільова функція квадратична, а умови лінійні – квадратичного програмування; цільова функція та умови – лінійні комбінації функцій однієї змінної – сепарабельного програмування; цільова функція та умови подані у вигляді поліномів – геометричного програмування.

4.3 Багатокритеріальна оптимізація

На практиці часто виникає випадок, коли замість однієї цільової функції $f(x)$ задано декілька цільових функцій $f_1(x),...,f_R(x)$. Така задача багатокритеріальної оптимізації має декілька постановок. В одній з них потрібно оптимізувати один з критеріїв, припустимо, $f_1(x)$, причому решту критеріїв утримують в заданих межах: $a_i \le f_i(x) \le b_i$ $(i = 2,3,...,k)$. В цьому разі фактично йдеться про звичайну багатокритеріальну оптимізацію. Що ж до нерівностей, які обмежують інші критерії, то їх можна розглядати як додаткові обмеження на припустиму область M.

В іншому випадку постановка полягає в упорядкуванні заданої множини критеріїв та послідовній оптимізації за кожним з них. Інакше, якщо проводять оптимізацію за першим критерієм $f_1(x)$, то одержують деяку множину $M_1 \subset M$, на якій функція $f_1(x)$ приймає оптимальне (екстремальне) значення. Прийнявши його за нову допустиму множину, проводять оптимізацію за другим критерієм та одержують в результаті нову допустиму множину $M_2 \subset M_1$. Якщо продовжити цей процес, то можна одержати після оптимізації за останнім критерієм $f_R(x)$ множину

M_R, яка і буде кінцевим результатом багатокритеріальної оптимізації. Звідси, якщо на деякому кроці i $(i<k)$ множина M_1 зведеться до однієї точки, процес оптимізації можна буде закінчити, оскільки $M_i = M_{i+1} = ... = M_k$. Зрозуміло, що, як і у випадку звичайної однокритеріальної оптимізації, задача може взагалі не мати розв'язку.

Третя постановка застосовує процес зведення багатьох критеріїв до одного за рахунок введення апріорних вагових коефіцієнтів λ_i для кожного з критеріїв $f_i(x)$. Як такі коефіцієнти можуть бути вибрані будь-які дійсні числа. Їх значення вибирають, виходячи з інтуїтивного подання ступеня важливості різних критеріїв: більш важливі критерії одержують ваги з більшими абсолютними значеннями. Після встановлення ваг λ_i багатокритеріальна задача зводиться до однокритеріальної з цільовою функцією $f(x) = \lambda_1 f_1(x) + ... + \lambda_R f_R(x)$.

Замість простої лінійної комбінації вхідних критеріїв можуть використовуватися і більш складні способи формування з них нового критерію.

4.4 Гладка оптимізація

У випадку, коли функція цілі $f(x)$ і функції $p_i(x)$, які задають обмеження, є диференціальними (гладкими), для розв'язання задач оптимізації використовується поняття градієнта. Для будь-якої функції $g(x)$, що диференціюється, її градієнтом $\nabla g(x)$в точці x називається вектор

$$\nabla g(x) = \begin{pmatrix} \dfrac{\partial g(x)}{\partial x_1} \\ \dots \\ \dfrac{\partial g(x)}{\partial x_n} \end{pmatrix}.$$

Вектор градієнта $\nabla g(x)$ задає (в даній точці x) напрям найшвидшого росту функції $g(x)$, а зворотний йому напрям $-\nabla g(x)$, що називається антиградієнтом – напрям найшвидшого спадання цієї функції. Точки, в яких градієнт функції перетворюється в нуль, називаються її стаціонарними точками. Якщо екстремум функції $f(x)$ досягається всередині припустимої області (не на її межі), то в точці оптимуму $x = a$ її градієнт перетворюється в нуль, тобто має місце система рівнянь

$$\left.\frac{\partial f(x)}{\partial x_1}\right|_{x=a} = 0, \ldots, \left.\frac{\partial f(x)}{\partial x_n}\right|_{x=a} = 0.$$

Ці рівняння (умови стаціонарності функцій) мають місце не тільки для абсолютних, але й для локальних екстремумів. Разом з тим умови стаціонарності не є достатніми.

4.4.1 Умови Куна-Таккера

У випадку, коли екстремальна точка a лежить на межі припустимої множини, вона не обов'язково є стаціонарною. Але за деякої додаткової умови можна, якщо замінити цільову функцію $f(x)$ так званою функцією Лагранжа $L(x) = f(x) + \sum_{i+1}^{m} \lambda_i p_i(x)$, (де $p_i(x)$ – ліві частини всіх граничних умов), домогтися того, щоб граничні екстремальні точки функції $f(x)$ були стаціонарними точками функції Лагранжа $L(x)$ при відповідних значеннях параметрів λ_i.

Умова, про яку йдеться, є так званою умовою регулярності – виконується на практиці. Вона полягає в лінійній незалежності в даній точці $x = a$ градієнтів всіх мережних функцій $p_i(x)$, для яких $p_i(a) = 0$.

При виконанні цієї умови для будь-якої точки максимуму x^* в задачі «знайти max $f(x)$ при $p_i(x) = 0$, $(i = 1, \ldots, l)$, $p_i(x) \geq 0$, $(i = l+1, \ldots, m)$» і для будь-якої точки мінімуму x^* в задачі «знайти min $f(x)$ при $p_i(x) = 0$, $(i = 1, \ldots, l)$, $p_i(x) \leq 0$, $(i = l+1, \ldots, m)$» повинні виконуватися такі три умови, які називаються *умовами Куна-Таккера:*

1) x^* лежить в припустимій множині;

2) $\lambda_i p_i(x^*) = 0$ при $i = 1, \ldots, m$;

3) $\nabla f(x^*) + \sum_{i+1}^{m} \lambda_i \nabla p_i(x^*) = 0$,

де λ_i – деякі дійсні числа (множники Лагранжа), довільні для всіх умов типу рівності $(i = 1, \ldots, l)$ і невід'ємні для всіх умов типу нерівності $(i = l+1, \ldots, m)$.

4.4.2 Чисельні методи гладкої оптимізації

Метод оптимізації, який використовує умови Куна-Таккера, на практиці застосовується відносно рідко через велику складність аналізу системи співвідношень, що виникає. Найчастіше застосовуються чисельні

методи оптимізації, для яких достатньо вміти знаходити чисельні значення градієнта в будь-якій заданій точці.

Загальна ідея методу градієнтного спуску (підйому)

Нехай неперервно диференційовна цільова функція $f(x)$ задана в усіх точках простору X. Вирушаючи від точки x на дуже малий крок ε в будь-якому напрямку d, де d – одиничний вектор, в силу умови диференційовності функції $f(x)$ одержимо нерівності:

$$f(x+\varepsilon\, d) \geq f(x) \geq f(x-\varepsilon\, d).$$

Це означає, що рух (на дуже малий крок) в напрямку градієнта функції забезпечує найбільший підйом, а в напрямку антиградієнта – найбільше спадання цієї функції. Ці напрямки називають, відповідно, напрямком найшвидшого підйому і найшвидшого спуску функції $f(x)$ в даній точці x.

Виходячи з заданої точки x^0, будуємо послідовність точок $(x^0, x^1, x^2, ...)$ так, що переміщення від кожної точки x^{i-1} до наступної точки x^i проводиться в напрямку найшвидшого підйому (при пошуку максимуму) або найшвидшого спуску (при пошуку мінімуму).

Важкість практичного застосування такої процедури полягає перш за все у виборі величини кроку ε_i при переході від точки x^{i-1} до точки x^i. Справа в тому, що нерівності мають місце лише в малому околі точки x, тобто для малої величини кроку ε_i. При великому значенні кроку можна, прямуючи в напрямку найшвидшого підйому, одержати не збільшення, а зменшення значення цільової функції $f(x)$. Те ж саме має місце і для процедури спуску.

Отже, величину кроку ε_i в процедурах найшвидшого підйому і спуску необхідно вибирати достатньо малою. Проте при прямуванні малими кроками для наближення до екстремальної точки може знадобитись дуже велика кількість кроків. Крім того, від процедур підйому і спуску звичайно потребується не просто наближення до екстремальної точки $x = a$, а збіжність до неї. Це означає, що зі збільшенням кількості кроків i відстань $|x^i - a|$ від точки x^i до точки екстремуму $x = a$ повинна наближатися до нуля. Це можливо лише в тому випадку, якщо з наближенням до точки екстремуму величина кроку ε_i буде наближатись до нуля. На рисунку 4.2 наведений алгоритм пошуку екстремуму функції за методом градієнтного спуску.

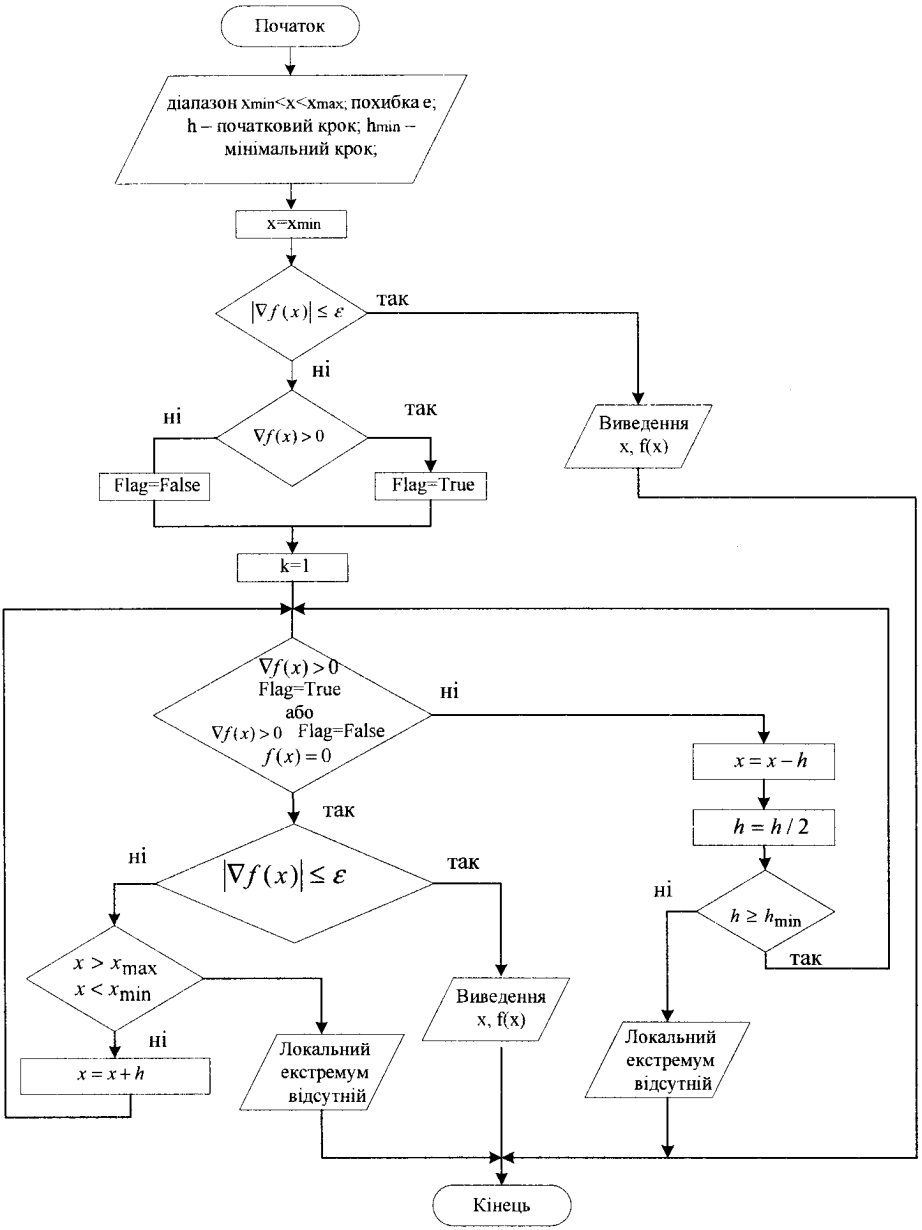


Рисунок 4.2 – Алгоритм методу градієнтного спуску (підйому)

Пропорційно-градієнтний метод

Найбільш простим способом забезпечення необхідного зменшення кроку при наближенні до точки екстремуму для диференціальної функції $f(x)$ є вибір довжини кроку ε_i пропорційним довжині вектора градієнта в точці x^{i-1}:

$$\varepsilon_i = \alpha\left|\nabla f(x^{i-1})\right|,\ \alpha > 0,\ \alpha = const.$$

Такий вибір кроку означає, що перехід від точки x^{i-1} до точки x^i буде виконуватись за формулою $x^i = x^{i-1} + \alpha\nabla f(x^{i-1})$ для процедури підйому (знаходження максимуму функції $f(x)$) і за формулою $x^i = x^{i-1} - \alpha\nabla f(x^{i-1})$ для процедури спуску (знаходження мінімуму функції $f(x)$).

Повнокроковий градієнтний метод

В цьому методі кожний крок градієнтного підйому або спуску виконується максимально можливої довжини, яка забезпечує необхідний напрямок зміни значення функції (тобто її збільшення або зменшення). Інакше кажучи, на напівпрямій, що виходить з чергової точки x^{i-1} в напрямку градієнта (при підйомі) або антиградієнта (при спусканні) відшукується точка абсолютного максимуму або мінімуму, яка і вибирається як наступна точка x^i.

У випадку довільної диференціальної функції $f(x)$ розв'язання подібної задачі, що її називають одновимірною оптимізаційною задачею, спрощується при накладанні на функцію $f(x)$ деяких додаткових вимог, наприклад, опуклості.

Метод спряжених градієнтів

Для випадку, коли функція $f(x)$ – квадратний поліном, розроблена точна процедура, при повних кроках якої точки екстремуму досягаються з будь-якої початкової точки x^0 за n кроків (де n – розмірність простору). Напрямок α^{i+1} кроку від точки x^i до точки x^{i+1} задається в цьому методі такою рекурентною формулою:

$$a^{i+1} = \pm\nabla f(x^i) + \frac{\left|\nabla f(x^i)\right|^2}{\left|\nabla f(x^{i-1})\right|^2} a^i,\ i = 1,2,...,$$

де $|\nabla f(x)|$ означає довжину вектора $\nabla f(x)$.

Знак плюс вибирається в тому випадку, коли розглядається задача максимізації, а знак мінус – у випадку розв'язання задачі мінімізації. Такий же вибір знака здійснюється і для напрямку α^1 першого кроку $(i=0)$. Він збігається з напрямком $\pm\nabla f(x^0)$.

4.4.3 Методи зведення загальної задачі оптимізації до задачі без обмежень

Чисельні методи гладкої оптимізації, які описані вище, в чистому вигляді застосовуються звичайно для задач без обмежень. Задачі загального вигляду (тобто задачі з обмеженнями) зводять до задач без обмежень за рахунок зміни цільової функції. Нехай, наприклад, потрібно знайти максимум функції $f(x)$ при обмеженнях $p_i(x)\geq 0$ $(i=1, ...,m)$. В так званому методі штрафних функцій будується функція $P(x)$, яку називають штрафною функцією і яка всередині припустимої області M приймає нульове значення, а поза нею – від'ємне. Найчастіше за таку функцію вибирають функцію вигляду

$$P(x)=-\sum_{i=1}^{m}\left|min[\,p_i(x),0\right|^{l},\ l\geq 1,$$

після чого будують послідовність додатних чисел $r_1>r_2>...>r_k>...$, яка збігається до нуля.

При заміні цільової функції $f(x)$ функцією

$$F_k(x)=f(x)+\frac{1}{r_k}P(x),\ k=1,2,...,$$

розв'язують задачу без обмежень для цієї функції, починаючи з $k=1$. Якщо розв'язок x^k належить припустимій області, тобто якщо всі $g_i(x^k)\geq 0$, то x^k є розв'язком початкової задачі з обмеженням. В протилежному випадку замінюють k на $k+1$ і розв'язують задачу для нової цільової функції $F_k(x)$.

Алгоритм розв'язання такої задачі наведений на рисунку 4.3.

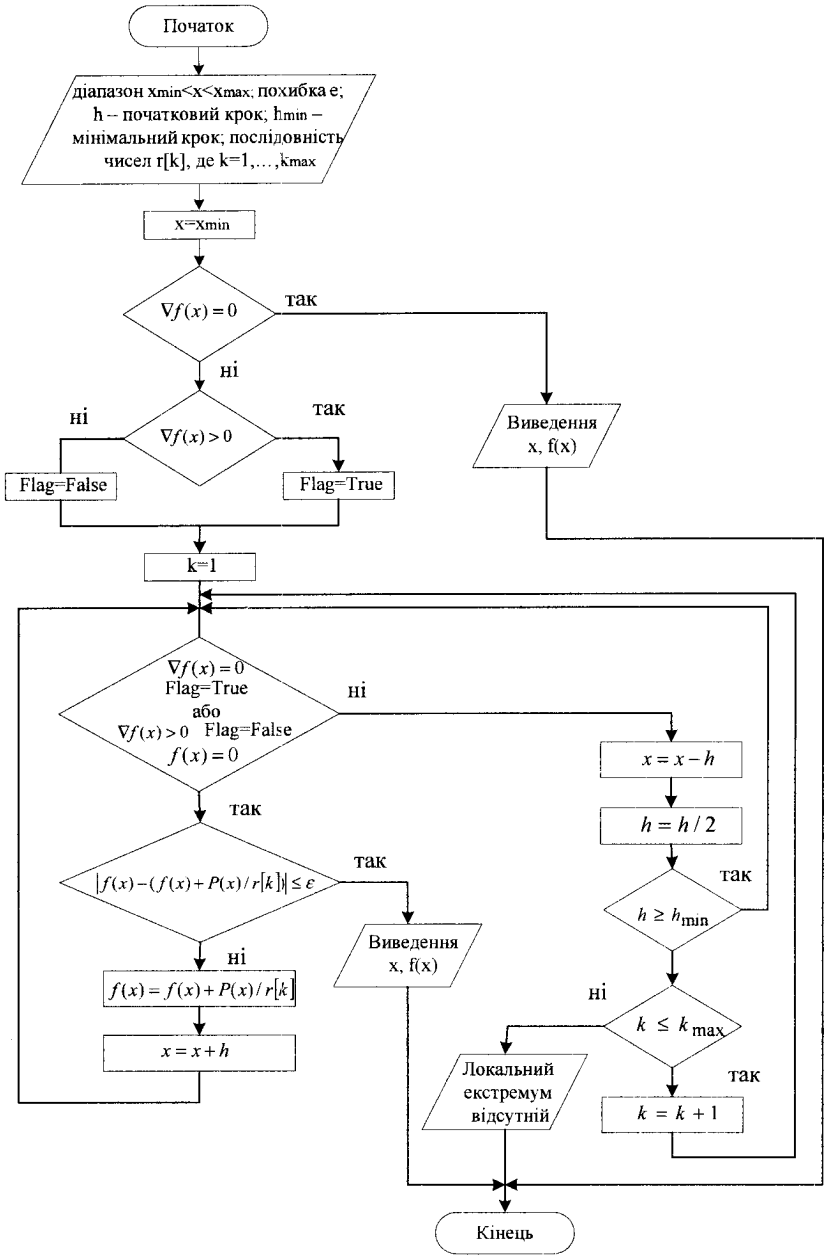


Рисунок 4.3 – Алгоритм методу штрафних функцій

В методі бар'єрів при розв'язанні задачі max $f(x)$ при $p_i(x) \geq 0$ $(i = 1,...,m)$ цільову функцію $f(x)$ замінюють функцією $F_k(x) = f(x) + r_k B(x)$, $r_k > 0$, де бар'єрна функція $B(x)$ характеризується властивістю прямувати до $-\infty$ при наближенні до границі припустимої області M зсередини. Таку функцію можна задати, наприклад, формулою:

$$B(x) = -\sum_{i=1}^{m} \frac{1}{p_i(x)},$$

якщо область M задана нерівностями $p_i(x) \geq 0$ $(i = 1, ..., m)$.

Таким же чином, як і в методі штрафних функцій, вибирають послідовність $\{r_k\}$ додатних чисел $r_1 > r_2 > ... > r_k > ...$, яка збігається до нуля. Розв'язуються задачі без обмежень для цільових функцій $F_k(x) = f(x) + r_k B(x)$ при $k = 1,2,...$

Якщо при цьому використовується той або інший чисельний метод, в якому початкова точка вибирається всередині M, то наявність бар'єра забезпечить належність екстремальної точки x^k (при її існуванні) області M. Наявність екстремуму забезпечується умовами обмеженості в області M і неперервності функцій $f(x)$ і $B(x)$.

4.5 Опукла оптимізація

Задача оптимізації може бути значно спрощена при накладанні на цільову функцію $f(x)$ і припустиму область M деяких обмежень. Одним з таких обмежень є так звана умова опуклості.

Множина M евклідового простору називається опуклою, якщо для будь-яких двох точок x^1 і x^2 цієї множини всі точки відрізка $[x^1, x^2]$, що їх з'єднує, також належать множині M.

Функція $f(x)$ опукла, або, точніше, опукла донизу, якщо вона визначена на опуклій множині M і для будь-яких двох точок x^1 і x^2 цієї множини значення $f(x)$ функції в будь-якій точці x відрізка $[x^1, x^2]$ не перевищують значення в тій же точці, що визначена на даному відрізку лінійної функції із значеннями $f(x_1)$ і $f(x_2)$ на його кінцевих точках. Це, в загальному випадку, виражається нерівністю

$$\alpha f(x^1) + (1-\alpha) f(x^2) \geq f(\alpha x^1 + (1-\alpha) x^2) \text{ при } 0 \leq \alpha \leq 1.$$

Очевидно, що при $0 \leq \alpha \leq 1$ вираз $\alpha x^1 + (1-\alpha) x^2$ задає різні точки відрізка $[x^1, x^2]$.

Для опуклої функції $f(x)$ в будь-якій точці x^0, що лежить в області визначення M функції, існує вектор $r(x^0)$, для якого при будь-якому x і M виконується нерівність $f(x)-f(x^0)\geq r(x^0);(x-x^0)$.

В правій частині цієї нерівності стоїть скалярний добуток векторів $r(x^0)$ і $x-x^0$, тобто добуток довжин цих векторів на косинус кута між ними. Вектор $r(x^0)$, що задовольняє цю властивість, називається узагальненим градієнтом (субградієнтом) функції $f(x)$ в точці x^0. Звичайний градієнт (у випадку його існування і відмінності від нуля) є також узагальненим градієнтом, причому єдиним, в розглядуваній точці. Якщо функція $f(x)$ недиференційовна в деякій точці x^0, в ній існує деяка множина узагальнених градієнтів, яку називають субградієнтною множиною функції $f(x)$ в точці x^0.

4.5.1 Простий субградієнтний метод опуклої оптимізації

Для реалізації субградієнтної оптимізації в випадку задачі без обмежень процес може починатися з будь-якої точки x^0. Зсув чергової точки x^i до наступної точки x^{i+1} здійснюється на відстань l_i в напрямку будь-якого узагальненого градієнта функції $f(x)$ в точці x^i у випадку задачі максимізації вгнутої функції і в зворотному напрямку – у випадку задачі мінімізації опуклої функції. Якщо кроки l_i зсувів обираються таким чином, що $l\to 0$ при $i\to\infty$, а ряд $l_0+l_1+l_2+...$ незбіжний, то послідовність $\{x^0, x^1, ...,x^m, ...\}$ збігається до множини екстремуму заданої функції $f(x)$ за умови, що множина обмежена.

Для обчислення субградієнтів застосовуються різні способи. Одним з них є зведення задачі обчислення субградієнта функції, що задана складним виразом, до обчислення субградієнтів окремих компонентів цього виразу. Для подібного зведення особливо часто застосовуються такі властивості опуклих функцій.

Властивість 1. Якщо функції $f_1(x), ...,f_m(x)$ опуклі, то опуклою буде і будь-яка їх лінійна комбінація $f(x)=\sum_{i=1}^{m}a_i f_i(x)$ з невід'ємними коефіцієнтами a_i $(a_i\geq 0,\ i=1,...,m)$, а субградієнт $S(x)$ функції $f(x)$ дорівнює лінійній комбінації $\sum_{i=1}^{m}a_i S_i(x)$ субградієнтів $S_i(x)$ функцій $f_i(x)$.

Властивість 2. Якщо функції $f_1(x),...,f_m(x)$ опуклі, то опуклою буде і функція $f(x)=max\, f_i(x)$, і всі субградієнтні множини $S_i(x^0)$ функцій $f_i(x)$, для яких $f(x^0)=f(x)$, входять в субградієнтну множину $S(x^0)$ функції $f(x)$ в будь-якій заданій точці $x=x^0$.

4.5.2 Методи розтягу простору

Простий субградієнтний метод, що розглянутий в попередньому пункті, може виявитися повільно збіжним, якщо області рівня цільової функції сильно витягнуті в одному напрямку, або, як часто при цьому кажуть, мають вигляд вузьких і довгих ярів. Ця ситуація характерна для багатьох задач негладкої оптимізації. Для поліпшення збіжності методу в подібних випадках використовують методи розтягу простору.

Розтяг простору зводиться до заміни змінних $x_1,...x_k$ в цільовій функції $f(x)=f(x_1,...,x_n)$ деякими їх лінійними комбінаціями. Особливо просто виконується розтяг в напрямку однієї з координатних осей. Змінна x_i, що відповідає цій осі, замінюється при цьому на $k^{-1}x_i$, де k – коефіцієнт розтягу, а решта залишається без змінювання.

4.6 Негладка оптимізація за методом координатного спуску (підйому)

При оптимізації недиференційовних функцій можна відмовитись від будь-яких узагальнень понять градієнта та використовувати лише кроки вздовж осей координат. З цією метою досліджують значення цільової функції $f(x_1,...,x_n)$ на черговому кроці оптимізації з приростом $\pm\delta_i$ однієї з координат і переходять від точки $(x_1,...,x_n)$ до точки $(x_1,...,x_{i-1},x_i\pm\delta_i,x_{i+1},...,x_n)$. Знак приросту вибирають таким чином, щоб значення функції $f(x_1,...,x_n)$ змінювалось в потрібному напрямку (збільшувалось при максимізації і зменшувалось при мінімізації). Величини δ_i вибирають звичайно як в покрокових методах та забезпечують максимально можливу зміну функції в обраному напрямку.

Зсуви здійснюються по черзі по всіх осях координат, причому після зсуву по останній осі x_n знову повертаються до першої осі x_1. Якщо повторити цей процес достатньо багато разів, як правило, можна знайти як завгодно точне наближення деякого екстремуму заданої негладкої функції в тому разі, коли виконується припущення, що такий екстремум існує.

4.7 Стохастична оптимізація

Іноді корисно замінити зсуви вздовж координатних осей зсувами (з відповідним знаком) вздовж випадково вибраного на кожному кроці напрямку. При достатньо загальних припущеннях внаслідок таких зсувів з імовірністю одиниця за достатньо велике число кроків екстремальна точка може бути апроксимована з будь-яким необхідним ступенем точності. Перевагою методу є простота кожного окремого кроку оптимізації. Проте кількість кроків порівняно з методами, які забезпечують пошук задовільних напрямків руху, у подібних простих методів, як правило, значно більша. При цьому на практиці вони можуть виявитися менш ефективними, ніж більш складні методи, що використовують поняття субградієнта і різні додаткові способи прискорення збіжності.

4.8 Лінійне програмування

Однією з задач опуклої оптимізації, що найчастіше зустрічаються, є так звана задача лінійного програмування, в якій як цільова функція, так і всі обмеження лінійні. Для розв'язання цієї задачі можуть бути застосовані загальні способи опуклої оптимізації, які описані в п'ятому підрозділі цього розділу. Але на практиці для цього випадку застосовують інші, більш прості способи, з одним з яких (так званий симплекс-метод) ми зараз ознайомимося.

Постановки задач лінійного програмування, які зустрічаються на практиці, передбачають ще одне додаткове обмеження. Це умова невід'ємності значень всіх змінних. Далі, обмеження типу нерівностей $p_i(x) \geq 0$ або $p_i(x) \leq 0$ введенням додаткових змінних з невід'ємними значеннями перетворюються в рівності $p_i(x) - z_i = 0$ або $p_i(x) + z_i = 0$. Крім того, простою зміною знака цільової функції задача знаходження максимуму зводиться до задачі знаходження мінімуму. Таким чином, в лінійному програмуванні можна обмежитись лише розв'язанням задачі мінімізації.

Стандартна постановка задачі лінійного програмування полягає в тому, щоб знайти мінімум лінійної функції $d + c_1 x_1 + c_2 x_2 + ... + c_n x_n$ при обмеженнях $b_i + a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n = 0 \ (i = 1,...,m), \ x_i \geq 0 \ (i = 1,...,m)$.

Інакше, йдеться про знаходження невід'ємного розв'язку $x = (x_1,...,x_n)$ даної системи лінійних алгебраїчних рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0 \ (i = 1,...,m)$, який перетворює в мінімум значення даної лінійної функції $d + \sum_{j+1}^{n} c_j x_j$.

Задача лінійного програмування може не мати розв'язку в таких випадках:

1) система S рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0 \ (i = 1,...,m)$ несумісна, тобто взагалі не має розв'язків;
2) система S не має жодного від'ємного розв'язку;
3) на множині M невід'ємних розв'язків системи S цільова функція $f = d + \sum_{j+1}^{n} c_j x_j$ може приймати як завгодно великі за абсолютною величиною від'ємні значення, тобто $\min_{M} f = -\infty$.

4.8.1 Симплекс-метод

Один з найбільш розповсюджених методів розв'язування задачі лінійного програмування в стандартній постановці полягає в послідовному застосуванні до лінійних функцій так званих симплексних перетворень, що являють собою розв'язання одного з обмежувальних рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0$ відносно деякого невідомого x_j, і підстановку знайденого таким чином значення x_j до всіх обмежувальних рівнянь і в цільову функцію $f = d + \sum_{j=1}^{n} c_j x_j$.

4.8.2 Транспортна задача

В деяких окремих випадках розв'язання задачі лінійного програмування може бути значно спрощено. Найважливішим з таких випадків є так звана транспортна задача, яку називають іноді задачею про призначення. Суть її полягає в мінімізації цільової функції

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_j$$

при $m + n$ обмеженнях типу рівнянь

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

Звичайно додатково припускається зберігання рівняння

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

інакше задача не буде мати розв'язків.

Назву транспортної задача, що розглядається, одержала тому, що до неї зводиться оптимізація плану перевезень вантажів з m пунктів відправлення з запасами $a_1,...,a_m$ до n пунктів призначення з об'ємами споживання $b_1,...,b_n$. Роль коефіцієнтів c_{ij} в цільовій функції виконують питомі вартості, тобто вартості перевезення однієї одиниці вантажу з пункту i в пункт j. Задача полягає в мінімізації загальної вартості перевезення вантажів за умови, що вантажі повністю вивезені з усіх пунктів відправлення і потреби всіх пунктів призначення виявились повністю задоволеними.

4.8.3 Цілочислове лінійне програмування

Іноді на практиці доводиться зустрічатися з такими задачами лінійного програмування, в яких припустимі лише цілочислові розв'язки.

В загальному випадку одержання цілочислового розв'язку потребує застосування спеціальних методів, серед яких найчастіше вживають методи відсікань та розгалужень і меж.

Методи відсікань базуються на таких ідеях. Спочатку розв'язується звичайна задача лінійного програмування A_0 (з тимчасово відкинутою вимогою цілочислового розв'язку). Якщо координати точки $a^0 = (a_1,...,a_n)$, яка одержана в результаті розв'язання задачі A_0, цілі числа, то ця точка дає розв'язок не тільки задачі A_0, але й початковій задачі цілочислового лінійного програмування, яку ми домовимося позначати через A.

Якщо хоча б одна з координат точки a^0, наприклад, a_j, не є цілим числом, то до початкової задачі цілочислового лінійного програмування додається нове обмеження, що побудоване з використанням інформації, яка є в таблиці симплекс-методу і відповідає кроку, на якому побудований розв'язок a^0.

В результаті виходить нова задача A_1, до якої знов застосовується та ж сама процедура. Таким чином, методи відсікань будують і розв'язують послідовність лінійних задач $A_0, A_1,...,$ кожна з яких відрізняється від попередньої лише одним новим обмеженням.

4.8.4 Загальна задача лінійної оптимізації

В загальному випадку задача дискретної оптимізації не припускає лінійності як цільової функції $f(x)$, так і обмежень $p_i(x) \geq 0$ $(i=1,...,m)$. Не вимагається також неодмінно цілочисельність координат точок, серед яких знаходиться розв'язок. Важливо лише, щоб вони утворювали дискретну множину M. Звичайно на практиці загальне число N точок виявляється настільки великим, що простий перебір фізично неможливий навіть при використанні найпотужніших ЕОМ. З цієї причини основна задача в дискретній оптимізації зводиться до розробки методів, які направлені на максимально можливе звуження перебору.

У так званих прямих методах дискретної оптимізації звичайно використовуються ті чи інші аналоги (в неперервному випадку) градієнтних методів, що розглядалися вище. До них відносять методи локальної оптимізації, наприклад, метод вектора спаду. Обчислювальну схему цього методу для розв'язання задачі мінімізації дійсної функції $f(x)$, що визначена на деякому просторі M дискретного метричного простору D, можна описати таким чином.

Спочатку вибираємо деяке початкове наближення $x^0 \in M$ і радіус r. В просторі D розглядаємо окіл $O(x^0,r)$ радіусом r з центром в точці x^0. Досліджуючи координати вектора спаду, який характеризує зміни значень цільової функції в точках множини $O(x^0,r) \cap M$ порівняно з $f(x^0)$, визначаємо, чи є x^0 точкою локального мінімуму функції f. Якщо ні, то за допомогою вектора спаду в множині $O(x^0,r) \cap M$ вибираємо точку x^1, для якої значення цільової функції менше, ніж $f(x^0)$. На наступній ітерації знову повторюємо цю ж процедуру, виходячи вже з x^1 як з центра нового околу, і так далі. Процес обчислень вважається закінченим, якщо одержано деякий локально оптимальний розв'язок задачі. Алгоритм методу припускає переривання обчислень на будь-якій ітерації з видачею як результату останнього одержаного припустимого розв'язку задачі.

В релаксаційних методах звертаються до способу ослаблення (релаксації) обмежень і заміни цільової функції $f(x)$ її мінорантою $f'(x)$ (в задачі мінімізації), тобто функцією, яка задовольняє умову $f'(x) \leq f(x)$ для всіх x.

З припустимої області M', що здобута розширенням вихідної області $M(M' > M)$ за рахунок ослаблення обмежень, M' і $f'(x)$ вибираються так, щоб, по-перше, релаксована задача допускала порівняно прості способи розв'язувань, і, по-друге, щоб вона найменше відрізнялась від вихідної за цільовою функцією і множиною припустимих розв'язків:

$$\min_{x \in M'} f'(x) \le \min_{x \in M'} f(x) \le \min_{x \in M} f(x)$$

Метод розгалужень і меж (МРМ)

Цей метод дозволяє отримати точний або наближений розв'язок з заданою відносною похибкою за цільовою функцією. Суть МРМ полягає в тому, що замість вихідної задачі оптимізації $A_0 = \{\min f_0(x), x \in M_0\}$ будується і розв'язується послідовність релаксованих задач $\overline{A_i} = \{\min \overline{f}_i(x), x \in \overline{M}_i\}$, $i = 0,1,2,...$ Спочатку розв'язується задача для M_0 і $\overline{f}_0(x_0) = f_0(x_0)$, то x_0 – оптимальний розв'язок A_0, і процес закінчується. Інакше $\overline{f}_0(x_0)$ – оцінка знизу для цільової функції на оптимальному розв'язку задачі A_0, і алгоритм починає процес розгалуження вихідної задачі A_0 на декілька задач (безпосередніх нащадків).

4.9 Теорія ігор

До методів дискретної оптимізації і лінійного програмування часто доводиться звертатись при аналізі і виборі рішень в конфліктних ситуаціях, тобто при наявності сторін, що мають на меті відмінні (найчастіше – протилежні) цілі.

Стратегією гравця називають сукупність правил, що визначають поведінку гравця від початку гри до її завершення.

Задання стратегій (A,B) двох гравців в парній грі повністю визначає її наслідок, тобто виграш одного і програш іншого. Гра називається скінченною, якщо в кожного гравця є лише скінченна кількість стратегій. Результати скінченної парної гри з нульовою сумою можна задавати матрицею, рядки і стовпці якої відповідають відмінним стратегіям, а її елементи – відповідні виграші однієї сторони (які дорівнюють програшам другої).

Розглянемо гру $m \times n$ з матрицею

	B_1	B_2	...	B_n
A_1	a_{11}	a_{12}	...	a_{1n}
A_2	a_{21}	a_{22}	...	a_{2n}
...	...	...	...	...
A_m	a_{m1}	a_{m2}	...	a_{mn}

Якщо перший гравець застосовує стратегію A_i, то другий буде прагнути до того, щоб вибором відповідної стратегії B_j звести виграш першого гравця до мінімуму. Величина цього мінімуму, яку ми позначимо α_i, дорівнює, очевидно, $\min\limits_j a_{ij}$.

З точки зору першого гравця (при будь-яких відповідях супротивника) доцільно прагнути знайти таку стратегію, при якій α_i перетворюється в максимум. Цей максимум, який ми позначимо α, називається нижньою ціною гри. Оскільки значення α обчислюється за формулою $\alpha = \max\limits_i \min\limits_j a_{ij}$, то його називають також максиміном. Йому відповідає максимінна стратегія, дотримуючись якої, перший гравець при будь-яких стратегіях противника забезпечить собі виграш, який не менше α (в залежності від знака α це може бути і програш, який в цьому разі виявиться мінімальним).

Аналогічно визначається мінімальний програш (який може бути в дійсності і виграшем) для другого гравця: $\beta = \min\limits_j \max\limits_i a_{ij}$.

Величина β називається верхньою ціною гри або мінімаксом. Їй відповідають мінімаксні стратегії другого гравця.

Має місце нерівність $\alpha \le \beta$. При $\alpha = \beta$ розв'язок одержується в чистих стратегіях. Для знаходження їх достатньо виділити в платіжній матриці максимінні (тобто рівні α) елементи. Будь-яка пара рядків і стовпців, на перетині яких знаходиться такий елемент, відповідає парі чистих оптимальних стратегій. При $\alpha < \beta$ перший гравець може істотно збільшити свій середній виграш порівняно з α, якщо він буде користуватись не чистою (однією єдиною) стратегією, а так званою змішаною стратегією. Змішана стратегія C полягає в тому, що при повторі гри здійснюється випадковий вибір стратегії з деякої множини змішуваних стратегій, і для кожної змішуваної стратегії вказується імовірність її вибору.

4.10 Динамічне програмування

На практиці часто доводиться зустрічатись з випадками, коли метою (ціллю) оптимізації є встановлення найкращої послідовності тих чи інших робіт (виробничих операцій, етапів будівництва різних споруд тощо). З подібною метою зустрічаються при розв'язанні задач так званого динамічного програмування.

Однією з перших задач такого роду, що привернули увагу математиків, була так звана задача про комівояжера (мандрівного торговця). Суть її така: є $n+1$ міст $A_0, A_1, ..., A_n$ $(n \ge 1)$ з заданими між ними відстанями d_{ij} $(i, j = 0,1,...,n)$. Потрібно, відправившись з A_0, вибрати такий маршрут

пересування $A_0, A_{i1}, A_{i2}, ..., A_{in}, A_0$, при якому комівояжер, побувавши в кожному місті по одному разу, повернувся б до вихідного пункту A_0, пройшовши при цьому мінімально можливий сумарний шлях.

Основний спосіб динамічного програмування полягає в знаходженні правил домінування, які дозволяють робити порівняння варіантів розвитку послідовностей і завчасне відсіювання безперспективних варіантів. У ряді випадків в задачах динамічного програмування вдається одержати такі сильні правила домінування, що вони визначають елементи оптимальної послідовності однозначно один за одним. В такому випадку правила домінування називають розв'язувальними правилами.

Розв'язувальні правила звичайно виводяться за допомогою принципу оптимальності Беллмана. Суть принципу оптимальності така. Нехай критерій F (задається формулою або алгоритмом), який дає числову оцінку якості варіанта (послідовності) $A_n = A_{i1}A_{i2}...A_{in}$, можна застосовувати не тільки до всієї послідовності, але і до будь-якого її початкового відрізка $A_R = A_{i1}A_{i2}...A_{iR}$. Послідовність A_n, якій відповідає екстремальне значення критерію F, називається оптимальною. Якщо будь-який початковий відрізок оптимальної послідовності також оптимальний (в класі всіх послідовностей, складених з тих же елементів, і можливо, такий, що має ті ж початок і кінець, що і даний відрізок), то вважають, що для відповідної задачі справедливий принцип оптимальності.

Розглянемо зразок розв'язання задачі про комівояжера методом динамічного програмування.

1. Введення даних про пункти $A_0, ..., A_n$ і відстані між пунктами i та j d_{ij} ($d_{ij}=0$ при $i=j$).

2. Обчислення всіх можливих варіантів відстаней, що складаються з трьох дільниць $A_0, A_{i1}, A_{i2}, A_{i3}$. Вони групуються за останнім пунктом i з них залишаються ті варіанти, що об'єднують однакові пункти, але мають найменший шлях.

3. До тих варіантів, що залишились, додають ще четверту дільницю і повторюють процедуру з пункту 2. Це повторюється для п'ятої, шостої і т. д. дільниць, доки не повертаються в пункт A_0. Той варіант (чи варіанти), що залишилися, і визначає найкоротший шлях, по якому комівояжеру можна об'їздити всі міста A_i ($i=0,...,n$), якщо він почне та закінчить свою подорож в A_0.

4.11 Варіаційні задачі

У варіаційних задачах місце змінних займають функції, а критерій оптимізації – деяка дійсна функція, що залежить від цих функцій і називається функціоналом.

В найпростішому випадку мають справу з функцією $y(x)$ однієї змінної і функціоналом вигляду

$$I = \int_{a_1}^{a_2} F(x, y, y')dx,$$

де a_1 і a_2 – константи, $F(x,y,z)$ – задана функція трьох дійсних змінних. При цьому припускається, що як сама функція $y = y(x)$, так і її похідна $y' = \frac{d}{dx} y(x)$ неперервні на відрізку $a_1 \le x \le a_2$.

В класичному варіаційному численні доводиться, що екстремум функції (мінімум або максимум) функціонала I досягається на функціях $y(x)$, які задовольняють диференціальне рівняння, тобто рівняння Ейлера:

$$F_y + F_{xy'} - F_{yy'}y' - F_{y'y'}y'' = 0.$$

Тут через F_y і F_{uv} ($u = x, y$ або y', $v = y'$) позначені, відповідно, перша і друга частинні похідні функції $F(x,y,y')$ за змінною y і за змінними u, v.

4.12 Системна оптимізація

В практиці проектування великих систем і управління такими системами, як правило, використовується багато критеріїв. В ряді випадків їх вдається в той чи іншій спосіб звести до одного критерію і тим самим повернутися до вже дослідженого випадку однокритеріальної оптимізації.

Найпростіший спосіб такого зведення – так зване зважування критеріїв. Якщо $f_1(x),...,f_n(x)$ функції, що виражають значення використовуваних критеріїв, то кожній з них, відповідно до відносної важливості критеріїв, вибирається додатний ваговий коефіцієнт λ_i. Операція зважування критеріїв (цільових функцій) $f_1(x),...,f_n(x)$ полягає в заміні їх єдиним критерієм (цільовою функцією) $f(x) = \lambda_1 f_1(x) + ... + \lambda_n f_n(x)$.

Але для багатьох задач, що пов'язані з великими системами, подібне зведення виявляється практично неможливим, так що в процесі оптимізації доводиться мати справу з векторною (багатокритеріальною) цільовою функцією. При цьому припустима область M може змінюватись в процесі оптимізації. Більше того, в її цілеспрямованій зміні як раз і полягає

основна змістовна сутність процесу оптимізації для подібного класу задач.

Наведемо одну з характерних формалізованих постановок задачі системної оптимізації.

Розв'язок відшукується безпосередньо в просторі K критеріїв оптимізації, які ми позначимо x_1 і x_2. Процес розв'язання починається з того, що в заданому просторі K вибирається деяка точка A_0 з координатами a_0, b_0 – бажаний розв'язок задачі. За цим будуються початкові обмеження $F_1^{(0)}(x_1, x_2) \geq 0, ..., F_n^{(0)}(x_1, x_2) \geq 0$, що задають початкову припустиму область P_0. Прямою перевіркою встановлюється, чи належить точка A_0 області P_0. В першому випадку в принципі може бути застосована звичайна (класична) процедура оптимізації або за одним з критеріїв x_1, x_2, або за тією чи іншою їх комбінацією.

Але при системному підході застосовується звичайно запропонований Л. С. Понтрягіним спосіб, а саме: відповідно до моделі M вищого рівня, що керує вибором критеріїв, точка A_0 виводиться з границь припустимої області P_0, як це показано на рисунку 4.4.

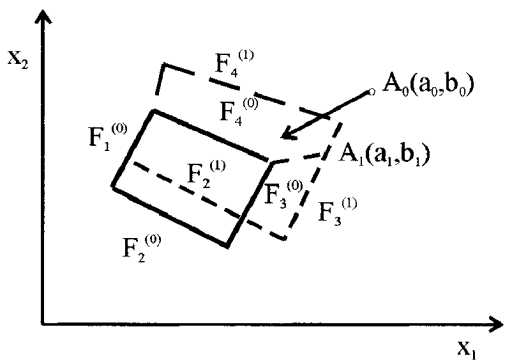


Рисунок 4.4 – Виведення точки A_0

Після цього виділяються ті обмеження, які не виконуються в точці A_0 (в випадку, що розглядається, ними будуть $F_3^{(0)}$ і $F_4^{(0)}$). Звертаючись до моделей M_3 і M_4, які формують ці обмеження, в діалоговому режимі опробовуються ті чи інші рішення, що змінюють відповідні обмеження в потрібному напрямку (якщо така зміна можлива). Оптимальним при цьому вважається той напрямок, що зменшує абсолютну величину від'ємних відхилень $F_i^{(0)}(a_0, b_0)$ (у випадку, який розглядається, $F_3^{(0)}(a_0, b_0)$ і $F_4^{(0)}(a_0, b_0)$).

4.13 Застосування теорії графів до розв'язання оптимізаційних задач

Розв'язання багатьох технічних задач можливо методами теорії графів. Основні поняття та визначення теорії графів розглянуті у відповідних підручниках, які є в списку літератури в кінці цього посібника. Тут ми обмежимося розглядом підходів до транспортної задачі та задачі комівояжера з застосуванням теорії графів.

Ці підходи базуються на відомій задачі з теорії графів про знаходження найкоротшого шляху між двома вершинами зв'язного неорієнтованого графу. До цієї задачі зводяться не тільки задача про комівояжера чи транспортна, а також, наприклад, багато задач оптимальної обробки деталей, оптимізації програмування, управління динамічними системами і т. д. В загальному вигляді задача формулюється так. Дано неорієнтований граф $G=(X,U)$, де X – множина вершин, U – множина ребер. Кожному ребру приписане деяке число $l(U)\geq 0$, що називається довжиною ребра (в транспортних задачах це може бути час чи вартість проїзду цим ребром). Треба для будь-яких вершин a і b графу G знайти такий шлях, що його довжина буде найкоротшою.

Загальне правило виявлення найкоротшого шляху в графі полягає в тому, щоб кожній вершині X приписати індекс λ_i, який дорівнює довжині найкоротшого шляху з даної вершини до кінцевої. Якщо, наприклад, спочатку взяти граф з ребрами одиничної довжини, то порядок дій буде такий:

1. Кінцевій вершині X_0 присвоюється індекс 0;
2. Усім вершинам, з котрих йде ребро до кінцевої вершини, присвоюється індекс 1;
3. Усім вершинам, що не мають індексів, та з котрих йде ребро в вершину з індексом λ_i, приписується індекс λ_i+1. Цей процес продовжується до тих пір, поки не буде помічена початкова вершина.

Після закінчення цього процесу індекс початкової вершини буде дорівнювати довжині найкоротшого шляху, а найкоротший шлях знаходиться, якщо пересуватися з початкової вершини в напрямку зменшення індексів.

Приклад наведено на рисунку 4.5. Подвійною лінією наведені два найкоротших шляхи.

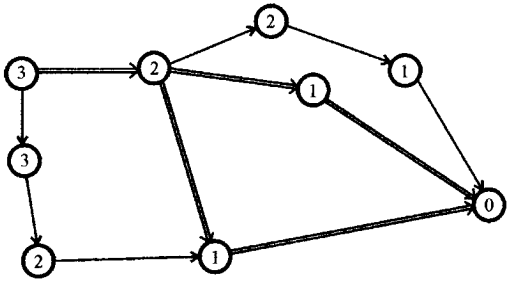


Рисунок 4.5 – Знаходження найкоротшого шляху в графі з ребрами одиничної довжини

Тепер розглянемо випадок, коли ребра мають довільну довжину. Процес присвоювання індексів ускладнюється, і порядок дій може бути зведений до такого:

1. Кожній вершині X_i присвоюється індекс λ_i. Спочатку кінцевій вершині X_0 присвоюється індекс $\lambda_0 = 0$. Для інших вершин беремо на першому кроці $\lambda_0 = \infty \ (i \neq 0)$;

2. Шукаємо таку дугу (x_i, x_j), що $\lambda_j - \lambda_i > l(x_i, x_j)$, й замінюємо індекс λ_j на індекс $\lambda_j = \lambda_i + l(x_i, x_j) < \lambda_j$. (Нагадаємо, що $l(x_i, x_j)$ – довжина дуги). Процес заміни індексів продовжується до тих пір, поки залишиться хоча б одна дуга, для якої можна зменшити λ_j.

Відзначимо дві важливі властивості індексів:

1. Нехай (X_k, X_s) – довільне ребро. Для нього обов'язково виконується умова $\lambda_s - \lambda_k \leq l(X_k, X_s)$. Це виходить з того, що при невиконанні умови індекс λ_s треба було б зменшити;

2. Нехай X_p – довільна вершина. При реалізації алгоритму присвоювання індексів індекс λ_p монотонно зменшується. Нехай X_q – остання вершина, що була використана для його зменшення. Тоді $\lambda_p = \lambda_q + l(X_q, X_p)$. Це приводить до висновку, що для будь-якої вершини X_p з індексом λ_p може бути знайдено вершину X_q, що з'єднується ребром з X_p, таку, що $\lambda_p - \lambda_q = l(X_q, X_p)$.

Виявлені властивості дають можливість сформулювати такий спосіб знаходження найкоротшого шляху.

Нехай $X_n = a$ – початкова вершина з індексом λ_n. Шукаємо вершину X_{p_1} таку, що $\lambda_n - \lambda_{p_1} = l(X_{p_1}, X_n)$. Далі шукаємо вершину X_{p_2} таку, що $\lambda_{p_1} - \lambda_{p_2} = l(X_{p_2}, X_{p_1})$, і т. д. до тих пір, поки не дійдемо до кінцевої

вершини. Шлях $\mu_0 = (X_n, X_{p_1}, ..., X_{p_k}, X_0)$ – найкоротший. Приклад використання цього способу поданий на рисунку 4.6. Найкоротший щлях виділений подвійною рискою.

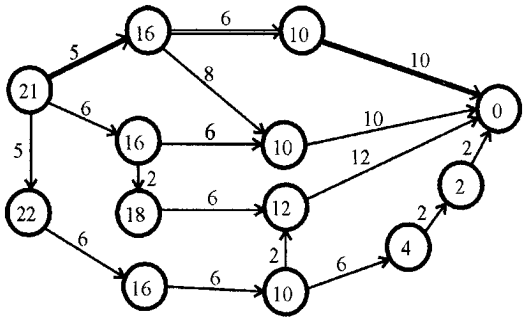


Рисунок 4.6 – Знаходження найкоротшого шляху в графі з ребрами довільної довжини

Очевидно, ці задачі про пошук найкоротшого шляху в графі аналогічні деяким задачам, що виникають в різних практичних областях. До них зводяться задачі про побудову доріг, що з'єднують декілька міст найдешевшим чином, задачі про розбудову електромережі, нафтопроводів та ін. За допомогою графових підходів можна розв'язати транспортну задачу та задачу про комівояжера. Розглянемо задачу про комівояжера у вигляді графу. Вершини цього графу відповідають містам, а шляхи – дорогам, що ці міста з'єднують. Кожній дузі можна приписати довжину, що дорівнює відстані між містами. І далі дуже просто звести задачу комівояжера до задачі про пошук найкоротшого шляху. До задачі про комівояжера зводиться багато транспортних задач, задач оптимізації програмування, оптимізації порядку оброблення деталей та ін.

4.14 Застосування активних експериментів при ідентифікації моделей

Перевагою пасивних методів перед активними є те, що при їх використанні не потребується спеціального втручання в хід технологічного процесу. Питання використання методів інтерполяції і апроксимації при обробці даних пасивних експериментів докладно розглянуті в 6 розділі першої частини посібника. Але ця перевага не компенсує недостатньої вірогідності математичних моделей, що отримуються. Тому в моделюванні розвинувся спеціальний напрямок – планування активного експерименту, що дозволяє ефективно використовувати наявні експериментальні ресурси.

Успішне проведення експерименту значною мірою залежить від

правильного вибору плану експерименту, який також визначає статистичний аналіз результатів. Вибір методу аналізу залежить від алгебраїчної моделі, що придатна для різних методів обробки даних, і від відомого або припустимого розподілу ймовірностей похибок вимірювань.

Як і в пасивному, в активному експерименті геометричним образом сукупності незалежних змінних $\overline{X}$ і залежної змінної y є простір $(n+1)$-го виміру, де n – число незалежних змінних; $(n+1)$-ий вимір стосується y. В цьому просторі залежності y від $\overline{X}$ відповідає n-вимірна поверхня, яку називають поверхнею відгуку (результат досліду розглядається як відгук системи на задану сукупність незалежних змінних або входів).

План експерименту вказує розташування дослідних точок в n-вимірному просторі незалежних змінних (або умови всіх дослідів, що їх треба провести). Найчастіше план експерименту задається у вигляді матриці планування – прямокутної таблиці, кожний рядок якої відповідає умовам певного досліду, а n стовпців – значенням однієї з незалежних змінних в різних дослідах. В $(n+1)$-му стовпці наводяться одержані в дослідах значення залежної змінної. Приклад матриці планування експерименту наведено в таблиці 4.1.

Таблиця 4.1 – Приклад матриці планування експерименту

№ досліду	x_1	x_2	x_3	y
1	3	7	1	75
2	4	7	2	78
3	4	9	1	93
4	3	9	2	97

Тут в чотирьох дослідах досліджується вплив на вихід y трьох факторів-входів: x_1, x_2 і x_3.

При одній незалежній змінній план спрощений і найчастіше зводиться до рівномірного розподілу точок через рівні інтервали. В разі більш ніж трьох змінних $(n>3)$ план ускладнюється, і при його побудові прагнуть одержати деякі оптимальні властивості:

– максимальну точність (мінімальну дисперсію) в досліджуваній області при даній кількості дослідів або, при мінімальній кількості дослідів, задану точність;

– незалежність (некорельованість) оцінок впливу різних факторів.

Повний факторний експеримент

Найбільше розповсюджений такий порядок побудови планів: на першому етапі вибирається центр досліджуваної області (центр плану) і в нього переноситься початок координат; вибирається інтервал варіювання

за кожною змінною – відстань по даній осі від центра до експериментальної точки. Вибір центра плану і інтервалу варіювання лежить поза математичною теорією. Цей етап повинен вирішуватись експериментатором на підставі знання об'єктів.

На наступному етапі здійснюють операцію приведення (кодування) змінних. Вона полягає в тому, що всі координати центра плану порівнюють з нулем, а інтервали варіювання Δ_j за кожною змінною приймають за одиницю. Кодовані змінні зручні тому, що вся обробка результатів дослідів проводиться в стандартній формі, яка не залежить від конкретних умов задачі; це істотно спрощує обчислення. Перехід від некодованих (натуральних) значень змінних z_j до кодованих x_j і назад відбувається за формулою

$$x_j = (z_j - z_{j0})/\Delta_j$$

і відповідно

$$z_j = z_{j0} + x_j\Delta_j,$$

де z_{j0} – координата центра плану за змінною j.

Розглянемо в наведених координатах найбільш поширений план 1-го порядку для двох незалежних змінних x_1 і x_2. Розташування точок показано на рисунку 4.7. Точки розташовуються у вершинах квадрата, центр 0 якого збігається з центром плану, сторони паралельні осям і дорівнюють 2. Матриця планування має такий вигляд (табл. 4.2):

Таблиця 4.2 – Приклад матриці планування експерименту

№ досліду	x_1	x_2
1	+1	+1
2	+1	-1
3	-1	+1
4	-1	-1

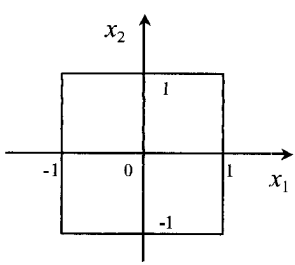


Рисунок 4.7 – План 1-го порядку для двох незалежних змінних

На практиці для скорочення запису часто замість «+1», «-1» пишуть «+», «-». Тоді матриця планування 1-го порядку буде мати вигляд (табл. 4.3):

Таблиця 4.3 – Приклад матриці планування експерименту

№ досліду	x_1	x_2
1	+	+
2	+	-
3	-	+
4	-	-

Основні особливості та властивості даного плану:

1. План побудований таким чином, що кожна незалежна змінна набуває в дослідах тільки два значення (+1 чи -1), тобто варіюється на двох рівнях (верхньому і нижньому). При цьому в чотирьох дослідах присутні всі можливі парні комбінації цих рівнів обох змінних. Чотири досліди, що поставлені за таким планом, називають повним факторним експериментом (ПФЕ) на двох рівнях для двох факторів; скорочено позначається ПФЕ 2^2;

2. При проведенні дослідів за планом ПФЕ розрахунок коефіцієнтів рівняння методом найменших квадратів спрощується, причому з цих дослідів можна знайти коефіцієнти не тільки лінійного рівняння регресії, але й такого, що містить ще один член;

3. Оптимальність використання експериментальних ресурсів: при заданих точності експерименту, кількості дослідів і межах вимірів факторів рівняння регресії виявляється більш точним, ніж те, що одержується з експерименту з іншим розташуванням досліджуваних точок;

4. Важлива особливість плану – некорельованість факторів. Вона базується на ортогональності матриці планування. Всі стовпці матриці планування (і весь план в цілому) ортогональні. З ортогональності однозначно випливає взаємна незалежність (некорельованість) всіх факторів і, відповідно, всіх коефіцієнтів рівняння регресії.

Вибір матриці планування у вигляді повного факторного експерименту при використанні як модель полінома першого степеня забезпечує оптимальне планування на підставі таких властивостей: всі розрахунки та обчислення проводяться просто; коефіцієнти регресії визначаються незалежно один від одного; дисперсії всіх коефіцієнтів регресії рівні і мінімальні; дисперсія вихідного параметра не залежить від обертання системи координат в центрі плану, а тільки від радіуса досліджуваної сфери факторного простору (властивість рототабельності).

Правильний вибір центра експерименту, інтервалів і рівнів варіювання факторів має вирішальне значення для достовірності побудованої математичної моделі. Основна вимога до інтервалу варіювання полягає в тому, щоб він не був меншим за подвійну квадратичну похибку фактора. Ця вимога пов'язана з тим, що інтервал між двома сусідніми рівнями

повинен значно впливати на вихідний параметр. Звичайно інтервал варіювання вибирається на підставі апріорної інформації, а потім уточнюється після одержання математичної моделі. Вдалий вибір інтервалу варіювання факторів гарантує достовірність математичної моделі об'єкта. Якщо інтервал варіювання вибраний невдало і модель неадекватна, то для уточнення моделі необхідно повторити експерименти.

Важливим елементом розробки плану експерименту є вибір числа рівнів для кожного фактора. Найбільше розповсюдження одержало планування факторів на двох рівнях, коли як рівні використовуються верхня і нижня межі інтервалу варіювання. Постановка експериментів за такими планами називається дворівневим повним факторним експериментом типу 2^n, де n – число факторів. Тоді для двох факторів число експериментів $2^2 = 4$, для трьох $2^3 = 8$ і т. д.

Розглянемо побудову матриці планування для повного факторного експерименту при трьох факторах (ПФЕ, 2^3). Щоб перебрати всі комбінації для трьох факторів на двох рівнях, випишемо два рази комбінації для двох факторів (ПФЕ, 2^2), один раз в поєднанні зі значенням $x_3 = +1$, другий – зі значенням $x_3 = -1$ (рисунок 4.8). Виходить такий план (матриця ПФЕ, 2^3).

Таблиця 4.4 – Приклад побудови матриці планування для повного факторного експерименту

№ досліду	1	2	3	4	5	6	7	8
x_1	+1	+1	-1	-1	+1	+1	-1	-1
x_2	+1	-1	+1	-1	+1	-1	+1	-1
x_3	+1	+1	+1	+1	-1	-1	-1	-1

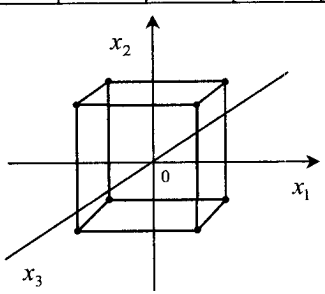


Рисунок 4.8 – Матриця планування для повного факторного експерименту

Приклад розробки математичної моделі за результатами активного експерименту розглянуто у Додатку А.

Дробовий факторний експеримент (ДФЕ)

Дворівневий план ефективний (особливо при $n>3$), коли для перевірки адекватності моделі мають достатню кількість ступенів вільності, тобто є перевищення числа дослідів N над числом коефіцієнтів моделі $(n+1)$, що визначаються. Число 2^n визначає обсяг вибірки при повному факторному експерименті. При певній постановці задачі іноді параметри доводиться варіювати не на двох, а на більшій кількості рівнів. Якщо число потрібних рівнів позначити через K, то обсяг повного факторного експерименту визначиться виразом K^n. Великий обсяг вибірки дає можливість точно визначити коефіцієнти моделі і мати достатньо ступенів вільності для перевірки адекватності моделі. Але це призводить до великих експериментальних і обчислювальних робіт. Наприклад, якщо в чотирифакторному експерименті фактори A і B мають по два рівні, фактор C – три і фактор D – чотири рівні, то число точок плану дорівнює $2*2*3*4=48$.

При збільшенні числа факторів і числа рівнів кожного фактора істотно зростає число точок плану, що створюють повну факторну решітку. Наприклад, якщо б кожний з семи факторів в деякому модельному прикладі мав тільки два рівні, то для побудови повного факторного плану було б потрібно $2^7=128$ точок. Таким чином, повний факторний план призводить до надмірних витрат часу і засобів.

В разі неповного дослідження (наприклад, дослідження лише головних ефектів і двофакторних взаємодій без розгляду взаємодій більш високого порядку) використовують неповні факторні плани, які дозволяють розв'язати поставлену задачу і потребують меншої кількості дослідів порівняно з повним факторним планом.

В будь-якому плані з числом дослідів меншим, ніж у повного факторного плану, присутні ефекти змішування.

Звичайно в ході експерименту цікавими є головні ефекти, і важливо, щоб вони не змішувались між собою. Практично в усіх застосовуваних неповних факторних планах головні ефекти змішуються лише з ефектами взаємодії високого порядку. Таким чином, при оцінюванні головних ефектів за допомогою цих планів припускається (в крайньому разі, попередньо), що взаємодії, які можуть змішуватися з головними ефектами, дорівнюють нулю або достатньо малі.

Кількість експериментів можна скоротити, якщо заздалегідь домовитись про структуру моделі. Припустимо, для досліджуваної задачі достатньо мати лінійну модель об'єкта. Тоді в разі наявності трьох змінних повний факторний експеримент дозволяє знайти вільний член, три коефіцієнти при лінійних членах і чотири ступені вільності для перевірки адекватності. При $m=6$ загальний обсяг експериментів дорівнює 64, з них

сім експериментів використовується для визначення вільного члена і коефіцієнтів при лінійних членах, а 57 – для перевірки адекватності моделі. В цьому випадку, як ми бачимо, є надлишок інформації для знаходження адекватності моделі процесу. При цьому доцільно ставити не всі 2^m експериментів, а менше, в залежності від вибраного числа ступенів вільності визначення адекватності моделі і від вибраної структурі моделі. Зменшити число експериментів можна, якщо прирівняти рівні однієї або декількох змінних добутку рівнів інших змінних, коли їх мінімальні та максимальні рівні прийняті, відповідно, за -1 і +1.

Нехай є чотири змінних. Повний факторний експеримент для такої кількості змінних дорівнює 16. Для лінійної структури моделі достатньо п'яти експериментів. При цьому рівень четвертої змінної можна брати за знаком добутку перших трьох змінних, тобто $x_4 = x_1 * x_2 * x_3$, де x_1, x_2, x_3 можуть приймати значення -1 чи +1. Якщо перший експеримент ставиться таким чином, що перша змінна знаходиться на нижньому рівні, тобто дорівнює -1, друга також на нижньому рівні, а третя – на верхньому, то четвертий параметр ставиться на верхній рівень. Число експериментів в такому випадку скорочується до восьми. Вісім експериментів достатньо для визначення вільних членів, коефіцієнтів при лінійних членах і для оцінювання адекватності моделі.

Такий план називається дробовою реплікою (ДР) від повного факторного експерименту. В залежності від постановки задачі повний факторний експеримент можна скорочувати до дробових реплік різного розміру. При наявності семи параметрів повний факторний експеримент 2^7, наприклад, можна скоротити до дробової репліки розміру 2^3, якщо прийняти, що $x_4 = x_1 * x_2$; $x_5 = x_1 * x_3$; $x_6 = x_2 * x_3$; $x_7 = x_1 * x_2 * x_3$.

В загальному випадку при наявності m параметрів і скороченні повного факторного експерименту на k ступенів вийде дробовий факторний експеримент обсягом 2^{m-k}.

Для побудови дробових реплік використовують спеціальні алгебраїчні вирази, які полегшують виявлення змішаних ефектів. Їх називають генерувальними співвідношеннями чи визначальними контрастами. Генерувальним називається співвідношення, що показує, які взаємодійні фактори замінені новими (воно генерує, або створює ДР).

З генерувальними співвідношеннями можна проводити алгебраїчні операції: множити обидві частини рівняння на будь-які ефекти – лінійні, подвійні, потрійні та інші взаємодії. При цьому фактор, що піднесений до квадрата або до іншого парного степеня, замінюють одиницею ($x_j^{2\nu} = 1;\ \nu = 1,2,3,\ldots$). Помноживши генерувальні співвідношення для плану 2^{3-1} на x_3, одержуємо $x_3^2 = x_1 * x_2 * x_3$; $x_3^2 = -x_1 * x_2 * x_3$, або, якщо врахувати вищесказане, $x_1 * x_2 * x_3 = 1$; $-x_1 * x_2 * x_3 = 1$.

Такі добутки називають визначальним контрастом. Визначальний контраст дає можливість встановити розв'язувальну здатність дробової репліки, тобто знайти, які з коефіцієнтів є незмішаними оцінками факторів. За прийнятим визначальним контрастом одержують співвідношення, які задають змішані оцінки для даної дробової репліки. Для цього кожний фактор множать на визначальний контраст.

Коефіцієнти рівняння регресії для ПФЕ і ДР визначаються за однаковими формулами. При використанні вимірних змінних x_1 переходять до безрозмірних (нормованих) змінних:

$$\bar{x}_i = (x_i - x_i^*)/\Delta x_i .$$

Тут $\bar{x}_i$ визначені через змінну на вихідному (основному) рівні +1. Для повних факторних планів (ПФП) і їх ДР

$$\sum_{u=1}^{n} \bar{x}_{iu}\bar{x}_{ju} = 0,\ j <> i;$$

$$\sum_{u=1}^{n} \bar{x}_{0u}\bar{x}_{iu} = \sum_{u=1}^{n} \bar{x}_{iu} = 0;$$

$$\sum_{u=1}^{n} (\bar{x}_{iu})^2 = n .$$

При використанні нормованих змінних спрощується розрахунок коефіцієнтів b та їх дисперсії. Для ПФП і ДР знаходимо

$$b_0 = \frac{y_0}{n};\ b_i = \frac{y_i}{n};\ b_{ij} = \frac{y_{ij}}{n};$$

$$\sigma_{bi}^2 \approx s_{bi}^2 = \frac{s^2}{n} = \frac{\sum_{u=1}^{n} (y_{uэ} - y_{up})^2}{\nu\ n},$$

де надійний інтервал $b_i = s_{bi} * t(p);\ \nu = n - (p+1)$.

Наведені співвідношення характеризують нормалізоване рівняння регресії $y = b_0 + b_1\bar{x}_1 + ... + b_p\bar{x}_p$.

Активна ідентифікація нелінійних моделей

Якщо математична модель процесу нелінійна, то, як і при пасивних методах, вона апроксимується поліномом. Однак при одержанні математичної моделі кількість необхідних дослідів при зростанні числа членів цього полінома швидко зростає. В зв'язку з цим потрібно інакше розв'язувати питання про число рівнів, центр плану експерименту і принципи оптимальності застосовуваних планів. Розв'язання цих питань здійснюється різними методами. Найбільш в інженерній практиці для опису нелінійної області використовуються методи центрального композиційного рототабельного та ортогонального планування. При плануванні за допомогою цих методів необхідно перш за все вибрати нульову точку, число рівнів і принцип оптимальності.

За нульову точки приймається центр плану, що вже був використаний раніше (наприклад, ПФЕ), який дає неадекватну лінійну модель (значущими виявляються ефекти взаємодії та квадратичні). При цьому проведені експерименти доповнюються спеціальними дослідами (такі плани називаються центральними композиційними). При використанні математичної моделі у вигляді полінома другого порядку двома рівнями варіювання факторів обмежуватись не можна. Але плани на трьох рівнях неекономічні з точки зору кількості дослідів. Якщо доповнити дворівневий план ПФЕ визначеними точками факторного простору, можна одержати оптимальний план. Ядро центрального композиційного плану складає ПФЕ типу 2^n при $n<5$. Якщо $n>5$, то користуються дробовими репліками, які забезпечують роздільне визначення лінійних ефектів і ефектів взаємодії. План ПФЕ доповнюють деякою кількістю зіркових точок, координати яких залежать від прийнятого принципу оптимальності. Загальна кількість дослідів при такому плануванні визначається формулою $N = 2^n + 2n + n_0$, де доданки – відповідно, число дослідів ПФЕ, зіркових і нульових точок.

Розглянемо побудову найбільш розповсюдженого типу планів другого порядку – центрального композиційного плану: центральні – внаслідок симетричності відносно центра плану, композиційні – оскільки вони компонуються додаванням певного числа дослідів до плану 1-го порядку. В цьому велика перевага таких планів: якщо рівняння 1-го порядку неадекватно описує об'єкт, то не треба ставити всі досліди заново, а достатньо додати точки – добудувати план до плану 2-го порядку, як показано на рисунок 4.9 ($n = 2$).

До точок повного факторного експерименту додаються точки в центрі плану (одна або декілька паралельних) і точки, що розташовані на всіх осях координат на відстанях $\pm\alpha$ від центра. Точки на осях називають зірковими точками, а величину α – зірковим плечем. Для різних планів зіркове плече може бути різним.

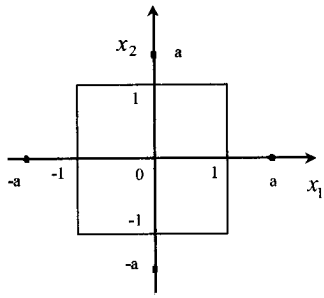


Рисунок 4.9 – Центральний композиційний план

Матриця такого плану для двох незалежних змінних (план 2-го порядку при $n = 2$) має вигляд:

Таблиця 4.5 – Приклад матриці планування експерименту

x_1	+1	+1	-1	-1	0^*	0^*	$+\alpha^*$	$-\alpha^*$	0^{**}
x_2	+1	-1	+1	-1	$+\alpha^*$	$-\alpha^*$	0^*	0^*	0^{**}

* – зіркова точка; ** – центральна точка

Загальне число точок плану 2-го порядку дорівнює $2^n + 2n + m = 9$, де $2n$ – число зіркових точок, m – число точок в центрі. Якщо план 1-го порядку – дробовий, то число точок плану 2-го порядку дорівнює $2^{n-k} + 2n + m$.

При рототабельному плануванні та повному факторному експерименті зіркове плече $\alpha = \frac{n}{2^4}$; для дробового факторного експерименту $\alpha = 2^{\frac{n-p}{4}}$, де p – дробовість репліки ($p = 1$ – півреплiка, $p = 2$ – чверть репліки і т. д.). Для вибору зіркового плеча, числа зіркових і нульових точок користуються даними таблиці 4.6.

Таблиця 4.6 – Дані для вибору зіркового плеча, числа зіркових і нульових точок

Параметри плану	Число змінних n				
	для ПФЕ типу 2^n				Для ДФЕ типу 2^{n-1}
	2	3	4	5	5
Число дослідів ядра матриці	4	8	16	32	16
Число зіркових точок	4	6	8	10	16
Число нульових точок	5	6	6	10	6
Зіркове плече	1,414	1,682	2,000	2,378	2,000

При рототабельному плануванні завдяки експериментальним точкам в центрі плану дисперсія передбачуваного значення всередині області експериментування постійна і не залежить від відстані до центра плану.

При обробці планів 2-го порядку потрібен значний об'єм обчислень, що їх виконують на ЕОМ.

При ортогональному плануванні до факторного експерименту або дробової репліки додають $2p+1$ дослідів (p – число змінних), причому один з них – «центральний» ($\bar{x}_1 = \bar{x}_2 = \ldots = \bar{x}_p = 0$), а в $2p$ – «зіркові». В «зіркових» дослідах кожна з нормованих змінних по черзі приймає значення $\pm\alpha$, а для решти змінних заданий основний рівень ($\bar{x}_i = 0;\ j <> i$). При кількості змінних $n = 2;\ 3\ i\ 4$ значення α_0 відповідно дорівнюють 1; 1,215 і 1,414.

За результатами ортогонального планування визначають коефіцієнти рівняння регресії:

$$b_i = \frac{\sum_{u=1}^{n} \bar{x}_{iu} y_u}{\sum_{u=1}^{n} (\bar{x}_{iu})^2}; \qquad b_{ij} = \frac{\sum_{u=1}^{n} (\bar{x}_{iu} \bar{x}_{ju}) y_u}{\sum_{u=1}^{n} (\bar{x}_{iu} \bar{x}_{ju})^2}; \qquad b_{ii} = \frac{\sum_{u=1}^{n} (\bar{x}_{iu})^2 y_u}{\sum_{u=1}^{n} (\bar{x}_{iu})^2};$$

$$b_0 = \frac{\sum_{u=1}^{n} y_u}{n} - b_{11} \frac{\sum_{u=1}^{n} (\bar{x}_{ju})^2}{n} - \ldots - b_{pp} \frac{\sum_{u=1}^{n} (\bar{x}_{pu})^2}{n}.$$

В ході оцінювання адекватності при рототабельному та ортогональному плануванні розрахункове значення критерію Фішера визначають як відношення дисперсії адекватності до дисперсії досліду:

$$F = \frac{s_{ad} / f_{ad}}{s_0 / f_0} = \frac{s_{ad}^2}{s_0^2}$$

і порівнюють з табличним F_T для ступенів вільності f_{ad} і f_0.

Контрольні запитання та завдання

1. Сформулюйте класичну постановку задачі оптимізації.
2. В чому полягає задача оптимального проектування?
3. За якими ознаками класифікуються задачі оптимізації?
4. Яка задача дістала назву математичного програмування?

5. Які особливості мають задачі нелінійного, стохастичного, динамічного, квадратичного, сепарабельного, геометричного програмування?
6. Чим відрізняються задачі багатопараметричної і багатокритеріальної оптимізації?
7. Сформулюйте умови стаціонарності для задачі гладкої оптимізації.
8. Запишіть та поясніть умови Куна - Такера.
9. Дайте характеристики та порівняйте чисельні методи розв'язання задачі гладкої оптимізації.
10. Спробуйте розв'язати задачу оптимізації іншими методами, окрім методу градієнтного спуску.
11. Як звести загальну задачу оптимізації до задачі без обмежень?
12. Які існують методи опуклої оптимізації?
13. Зробіть постановку задачі лінійного програмування. В яких випадках вона не має розв'язку?
14. Дайте характеристику методу розгалужень і меж розв'язання задачі лінійної оптимізації.
15. В яких випадках і як застосовується теорія ігор для розв'язання задач дискретної оптимізації? Як вибирається стратегія гравця?
16. Сформулюйте принципи оптимізації Беллмана.
17. Яку задачу називають задачею комівояжера?
18. Складіть програми пошуку найкоротшого шляху в графах з ребрами одиничної та довільної довжини.
19. Чим відрізняються активні та пасивні експерименти?
20. Що таке план експерименту? Як він задається? Чим відрізняються дробовий та повний факторні експерименти?
21. Як розраховуються коефіцієнти рівняння регресії?
22. Як визначити необхідну кількість випробувань при проведенні активних експериментів?

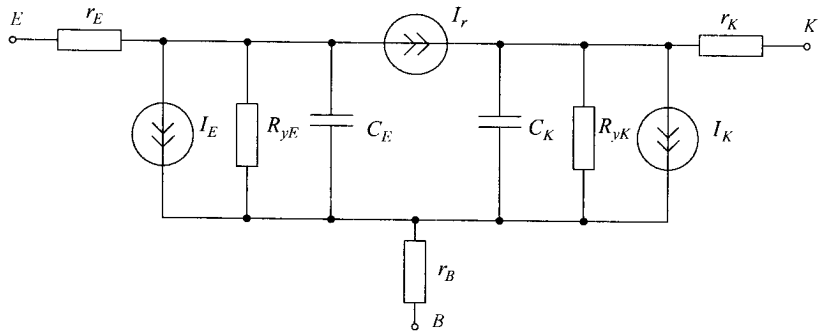
ДОДАТКИ

ДОДАТОК А

ЧИСЕЛЬНИЙ РОЗРАХУНОК ДЕЯКИХ ЗАДАЧ

Приклади до розділу 1 (частина 1). Математичне моделювання

Приклад 1.1. Математична модель біполярного транзистора. Розглянемо еквівалентну схему біполярного транзистора. Прийняті такі позначення: I_E, C_E, R_{yE}, I_K, C_K, R_{yK} – відповідно елементи *p-n* переходів емітер-база і колектор-база; $I_r = \beta I_E - \beta_I I_K$ – джерело струму, що відображає проліт неосновних носіїв через базу і визначає підсилювальні властивості транзистора (β і β_I – нормальний і інверсний коефіцієнти підсилення струму); r_E, r_K і r_B – об'ємні опору областей емітера, колектора і бази.



Запишемо компонентні рівняння для кожного елемента. Отримаємо таку систему рівнянь:

$$
\left.
\begin{aligned}
&I_{rE} - U_{rE}/r_E = 0;\\
&I_{RyE} - U_E/R_{yE} = 0;\\
&I_E - I_{TE}\exp(U_E/m\varphi_{TE} - 1) = 0;\\
&I_{CE} - [C_{БE} + \tau/(m\varphi_{TE})(I_E + I_{TE})]\dot{U}_{CE} = 0;\\
&I_r - \beta I_E + \beta_I I_K = 0;\\
&I_{CK} - [C_{БK} + \tau_П/m\varphi_{TK}(I_K + I_{TK})]\dot{U}_{CK} = 0;\\
&I_K - I_{TK}\exp(U_K/m\varphi_{TK} - 1) = 0;\\
&I_{RyK} - U_K/R_{yK} = 0;\\
&I_{rK} - U_{rK}/r_K = 0;\\
&I_{rБ} - U_{rБ}/r_B = 0,
\end{aligned}
\right\}
$$

де I_{TE} – тепловий струм переходу база-емітер; m – емпіричний коефіцієнт; φ_{TE} – температурний потенціал емітера; $C_{БЕ}$ – бар'єрна ємність переходу база - емітер; $C_{БК}$ – бар'єрна ємність переходу база-колектор; φ_{TK} – температурний потенціал колектора; I_{TK} – тепловий струм переходу база-колектор; $\tau_{П}$, τ – параметри, що характеризують час прольоту носіїв струму через області транзистора.

Невідомими змінними в цьому випадку є

$$I_{rE}, I_{RyE}, I_E, I_{CE}, I_r, I_{CK}, I_K, I_{RyK}, I_{rK}, I_{rБ}, U_{rE}, U_E, \dot{U}_{CE}, \dot{U}_{CK}, U_K, U_{rK}, U_{rB}.$$

З цього переліку виходить, що в моделі враховані деякі топологічні рівняння: вилучені величини $\dot{U}_{CE}$ і $\dot{U}_{CK}$, U_R і U_{rK} величини, які збігаються відповідно з величинами U_E і U_K.

Запишемо топологічні рівняння системи:

$$\left.\begin{aligned}
&I_{rE} - I_{RyE} - I_E + I_{CE} - I_r = 0;\\
&I_r + I_{CK} - I_K - I_{RK} - I_{rK} = 0;\\
&I_{RE} + I_E - I_{CE} - I_{CK} + I_K + I_{RyK} - I_{rB} = 0;\\
&U_E - U_K - U_r = 0;\\
&U_{rE} + U_E + U_{rБ} - U_{БЕ} = 0;\\
&-U_K + U_{rK} - U_{rБ} + U_{БК} = 0.
\end{aligned}\right\}$$

У двох останніх рівняннях $U_{БЕ}$ і $U_{БК}$ – величини напруги база-емітер і база-колектор. У систему вносять різницеві апроксимації для похідних $\dot{U}_{CE}, \dot{U}_{CK}$ з кроком h.

Таким чином, математична модель біполярного транзистора є системою рівнянь. Матриця Якобі для цієї системи наведена в таблиці А.1 (нульові елементи не позначені).

У цій матриці прийняті такі позначення коефіцієнтів:

$$a_1 = -\frac{I_{TE}}{m\varphi_{TE}} \exp\left[U_E/(m\varphi_{TE})\right];$$

$$a_2 = -\frac{\tau}{m\varphi_{TE}} U_{CE};$$

$$a_3 = -\left[C_{BE} + \frac{\tau}{m\varphi_{TE}}(I_E + I_{TE})\right];$$

Таблиця А.1 – Матриця Якобі для математичної моделі біполярного транзистора

№ рівняння	Змінна																	
	I_{rE}	$I_{Rн}$	I_E	I_{CE}	I_r	I_{CK}	I_K	$I_{Rн\hat{E}}$	I_{rK}	I_{rB}	U_{rE}	U_E	$U_{БЕ}$	$U_{БК}$	$U_{\hat{E}}$	U_{rK}	U_{rB}	U_r
1	1										$-\frac{1}{r_E}$							
2		1										$-\frac{1}{R_{нE}}$						
3			1									α_1						
4			α_2	1									α_3					
5			$-\beta$		1		β_1											
6						1	α_4							α_5				
7							1								α_6			
8								1							$-\frac{1}{R_{нK}}$			
9									1							$-\frac{1}{r_K}$		
10										1							$-\frac{1}{r_B}$	
11	-1	-1	-1	1	-1													
12					1	1	-1	-1	-1									
13		1	1	-1		-1	1	1		-1								
14												1			-1			-1
15											1	1					1	
16															-1	1	-1	
17												$-\frac{1}{h}$	1					
18														1	$-\frac{1}{h}$			

$$a_4 = -\frac{\tau_{II}}{m\varphi_{TK}} U_{CK};$$

$$a_5 = -\left[C_{BK} + \frac{\tau_{И}}{m\varphi_{TK}} (I_K + I_{TK}) \right];$$

$$a_6 = -\frac{I_{TK}}{m\varphi_{TK}} \exp[U_K / (m\varphi_K)].$$

Приклади до розділу 2 (частина 1)
Розв'язання систем лінійних рівнянь

Приклад 2.1. Розв'язати СЛАР методом Гаусса з вибором головного елемента

$$\begin{cases} 5 \cdot x_1 + 6 \cdot x_2 + 7 \cdot x_3 + 8 \cdot x_4 = 1 \\ 10 \cdot x_1 + 10 \cdot x_2 + 11 \cdot x_3 + 12 \cdot x_4 = 2 \\ 15 \cdot x_1 + 4 \cdot x_2 + 3 \cdot x_3 + 5 \cdot x_4 = 3 \\ 20 \cdot x_1 - x_3 - 2 \cdot x_4 = 4 \end{cases}$$

Розв'язання

Сформуємо матрицю коефіцієнтів A і стовпець B:

$$A = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 10 & 10 & 11 & 12 \\ 15 & 4 & 3 & 5 \\ 20 & 0 & -1 & -2 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Головний елемент матриці A дорівнює 20. При перестановці рядків отримаємо $a_{11} = 20$.

$$A = \begin{pmatrix} 20 & 0 & -1 & -2 \\ 10 & 10 & 11 & 12 \\ 15 & 4 & 3 & 5 \\ 5 & 6 & 7 & 8 \end{pmatrix} \qquad B = \begin{pmatrix} 4 \\ 2 \\ 3 \\ 1 \end{pmatrix}$$

Розрахуємо елементи матриці A^1 і стовпця B^1:

$$A^1 = \begin{pmatrix} 1 & -0.1 & -0.05 & 0 \\ 0 & 12 & 11.5 & 10 \\ 0 & 6.5 & 3.75 & 4 \\ 0 & 8.5 & 7.25 & 6 \end{pmatrix} \quad B^1 = \begin{pmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Головний елемент нової матриці $a_{22}^1 = 12$.

Аналогічно проведемо подальші обчислення:

$$A^2 = \begin{pmatrix} 1 & -0.1 & -0.05 & 0 \\ 0 & 1 & 0.96 & 0.83 \\ 0 & 0 & -2.5 & -1.4 \\ 0 & 0 & -0.91 & -1.06 \end{pmatrix} \qquad B^2 = \begin{pmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 1 & -0.1 & -0.05 & 0 \\ 0 & 1 & 0.96 & 0.83 \\ 0 & 0 & 1 & 0.56 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad B^3 = \begin{pmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Матриця A^4 має трикутний вигляд. Система має єдиний розв'язок. Для його пошуку застосуємо зворотний хід метода Гаусса.

$x_4 = 0;$

$x_3 = b_3^4 - a_{34}^4 \cdot x_4 = 0 - 0.56 \cdot 0 = 0;$

$x_2 = b_2^4 - a_{24}^4 \cdot x_4 - a_{23}^4 \cdot x_3 = 0 - 0.83 \cdot 0 - 0.96 \cdot 0 = 0;$

$x_1 = b_1^4 - a_{14}^4 \cdot x_4 - a_{13}^4 \cdot x_3 - a_{12}^4 \cdot x_2 = 0.2 - 0 - 0.05 \cdot 0 - 0.1 \cdot 0 = 0.2;$

Розв'язок: $X = \begin{pmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$

Приклад 2.2. Розв'язати СЛАР з прикладу 2.1 методом Крамера.

Розв'язання

$\det(A) = 210$

$$A_1 = \begin{pmatrix} 1 & 6 & 7 & 8 \\ 2 & 10 & 11 & 12 \\ 3 & 4 & 3 & 5 \\ 4 & 0 & -1 & -2 \end{pmatrix} \qquad \det(A_1) = 42; \quad x_1 = \frac{\det(A_1)}{\det(A)} = \frac{42}{210} = 0.2;$$

$$A_2 = \begin{pmatrix} 5 & 1 & 7 & 8 \\ 10 & 2 & 11 & 12 \\ 15 & 3 & 3 & 5 \\ 20 & 4 & -1 & -2 \end{pmatrix} \qquad \det(A_2) = 0; \quad x_2 = \frac{\det(A_2)}{\det(A)} = \frac{0}{210} = 0;$$

$$A_3 = \begin{pmatrix} 5 & 6 & 1 & 8 \\ 10 & 10 & 2 & 12 \\ 15 & 4 & 3 & 5 \\ 20 & 0 & 4 & -2 \end{pmatrix} \qquad \det(A_3) = 0; \quad x_3 = \frac{\det(A_3)}{\det(A)} = \frac{0}{210} = 0;$$

$$A_4 = \begin{pmatrix} 5 & 6 & 7 & 1 \\ 10 & 10 & 11 & 2 \\ 15 & 4 & 3 & 3 \\ 20 & 0 & -1 & 4 \end{pmatrix} \qquad \det(A_4) = 0; \quad x_4 = \frac{\det(A_4)}{\det(A)} = \frac{0}{210} = 0;$$

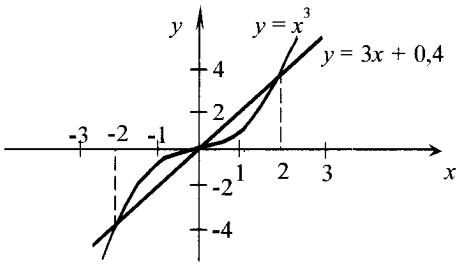
Розв'язок системи X:

$$X = \begin{pmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Приклади до розділу 3 (частина 1)
Нелінійні задачі

Приклад 3.1. Метод графічного відділення коренів.
Розглянемо як приклад рівняння $x^3 - 3x - 0{,}4 = 0$. Запишемо його як

$$x^3 = 3x + 0{,}4$$



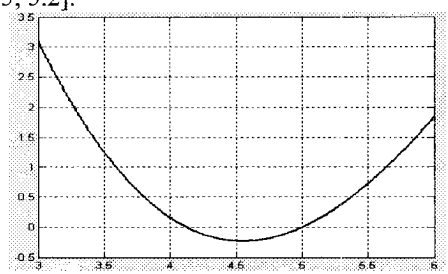
З графіка видно, що тут три корені: $c_1 \in [-2, -1]$; $c_2 \in [-1, 0]$; $c_3 \in [1, 2]$.

Приклад 3.2. Методом половинного ділення знайти розв'язок рівняння з точністю 0,05.

$$f(x) = (x-5)^2 + \sin(x-5) = 0\,.$$

Розв'язання

На основі побудованого графіка функції визначаємо відрізок, який містить корінь: [4.85; 5.2].



Крок 1

$a = 4.85;$ $\qquad b = 5.2;$

$$c = \frac{b+a}{2} = \frac{4.85 + 5.2}{2} = 5.025\,;$$

$$\varepsilon = |b - a| = |5.2 - 4.85| = 0.35 > 0.05;$$

$f(a) = f(4.85) = (4.85-5)^2 + \sin(4.85-5) = -0.127;$

$f(c) = f(5.025) = (5.025-5)^2 + \sin(5.025-5) = 0.026;$

$f(b) = f(5.2) = (5.2-5)^2 + \sin(5.2-5) = 0.239;$

$f(a) \cdot f(c) < 0 \Rightarrow b := c = 5.025;$

Крок 2

$a = 4.85;$ $\quad b = 5.025;$

$c = 4.938;$ $\quad \varepsilon = 0.175 > 0.05;$

$f(a) = f(4.85) = -0.127;$

$f(c) = f(4.938) = -0.059;$

$f(b) = f(5.025) = 0.026;$

$f(c) \cdot f(b) < 0 \Rightarrow a := c = 4.938;$

Крок 3

$a = 4.938;$ $\quad b = 5.025;$

$c = 4.981;$ $\quad \varepsilon = 0.088 > 0.05;$

$f(a) = f(4.938) = -0.059;$

$f(c) = f(4.981) = -0.018;$

$f(b) = f(5.025) = 0.026;$

$f(c) \cdot f(b) < 0 \Rightarrow a := c = 4.981;$

Крок 4

$a = 4.981;$ $\quad b = 5.025;$

$\varepsilon = 0.044 < 0.05;$

$c = 5.003;$

Таким чином, з заданою точністю знайдено корінь $c = 5.003$.

Приклад 3.3. Розглянемо застосування методу хорд для рівняння з прикладу 3.1.

Рівняння має вигляд:

$$f(x) = (x-5)^2 + \sin(x-5) = 0.$$

На основі побудованого графіка функції був визначений відрізок, який містить корінь [4.85; 5.2].

Таким чином

$a = 4.85; \qquad b = 5.2;$

Знайдемо перше наближення до кореня:

$$x_t = a - \frac{f(a)\cdot(b-a)}{f(b)-f(a)} = 4.85 - \frac{f(4.85)\cdot(5.2-4.85)}{f(5.2)-f(4.85)} = 4.972.$$

Визначимо питому частину відрізка

$$f(a)\cdot f(x_t) > 0; \qquad f(b)\cdot f(x_t) < 0;$$

$$\Rightarrow a := x_t = 4.972$$

Запам'ятовується останнє наближення $x_p := x_t = 4.972$.

Знайдемо наступне наближення:

$$x_t = a - \frac{f(a)\cdot(b-a)}{f(b)-f(a)} = 4.972 - \frac{f(4.972)\cdot(5.2-4.972)}{f(5.2)-f(4.972)} = 4.995.$$

Знову визначимо питому частину відрізка

$$f(a)\cdot f(x_t) > 0; \qquad f(b)\cdot f(x_t) < 0;$$

$$\Rightarrow a := x_t = 4.995.$$

Перевіряємо умову

$$|x_t - x_p| = |4.995 - 4.972| = 0.024 < 0.05.$$

Таким чином, знайдено наближений корінь $x = 4.995$. Аналізуючи кількість операцій, зазначимо, що розв'язок знайдено швидше, ніж методом половинного ділення.

Приклад 3.4. Розглянемо приклад застосування методу Ньютона для задачі з прикладу 3.2.

Рівняння має вигляд

$$f(x) = (x-5)^2 + \sin(x-5) = 0.$$

На основі побудованого графіка функції був визначений відрізок, який містить корінь [4.85; 5.2].

Таким чином

$a = 4.85; \qquad b = 5.2;$

$f'(x) = 2\cdot x - 10 + \cos(x-5); \quad f''(x) = 2 - \sin(x-5).$

Визначимо початкове наближення:

$f(a)\cdot f''(a) < 0; \qquad f(b)\cdot f''(b) > 0; \Rightarrow x_0 = 5.2.$

Знайдемо наступні наближення до кореня:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 5.2 - \frac{f(5.2)}{f'(5.2)} = 5.027;$$

$$\varepsilon = |x_1 - x_0| = 0.173 > 0.05;$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 5.027 - \frac{f(5.027)}{f'(5.027)} = 5.001;$$

$\varepsilon = |x_2 - x_1| = 0.026 < 0.05.$

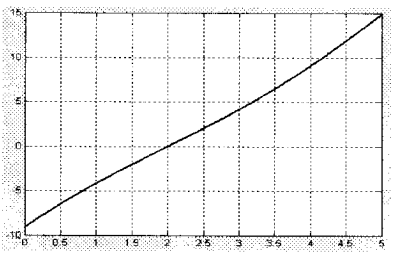
Таким чином, знайдено корінь $x = 5.001$.

Приклад 3.5. Методом ітерацій знайти нуль функції з точністю 0,05.

$$f(x) = (5 \cdot x - 10) - \sin(x - 2) \quad \varepsilon = 0.05$$

Розв'язання

Побудуємо графік функції.



З графіка функції знаходимо відрізок, який містить корінь рівняння [1.2; 2.2].

Таким чином

$a = 1.2$; $b = 2.2$;

Подамо рівняння у вигляді:

$$x = g(x) = \frac{1}{5} \cdot \sin(x - 2) + 2.$$

Початкове наближення візьмемо як середину відрізка:

$x_0 = 1.7 \in [a; b]$.

Знайдемо наступні наближення до кореня:

$$x_1 = g(x_0) = \frac{1}{5} \cdot \sin(x_0 - 2) + 2 = \frac{1}{5} \cdot \sin(1.7 - 2) + 2 = 1.941;$$

$x_1 = 1.941 \quad \varepsilon = |x_1 - x_0| = 0.241 > 0.05$;

$$x_2 = g(x_1) = \frac{1}{5} \cdot \sin(x_1 - 2) + 2 = \frac{1}{5} \cdot \sin(1.941 - 2) + 2 = 1.988;$$

$x_2 = 1.988 \quad \varepsilon = |x_2 - x_1| = 0.047 < 0.05$.

Отже, знайдено наближений нуль функції $x = 1.988$ з заданою точністю.

Приклад 3.6. Знайти корінь системи за допомогою методу Ньютона

$$\begin{cases} F(x,y)=2x^3-y^2-1=0; \\ G(x,y)=xy^3-y-4=0. \end{cases}$$

Розв'язання

Графічним шляхом можна знайти приблизно $x_0 = 1{,}2$ і $y_0 = 1{,}7$.

$$J(x,y)=\begin{vmatrix} 6x^2 & -2y \\ y^3 & 3xy^2-1 \end{vmatrix}.$$

В початковій точці

$$J(1{,}2;1{,}7)=\begin{vmatrix} 8{,}64 & -3{,}40 \\ 4{,}91 & 9{,}40 \end{vmatrix} = 97{,}910.$$

За формулами одержуємо

$$x_1 = 1{,}2-\frac{1}{97{,}910}\begin{vmatrix} -0{,}434 & -3{,}40 \\ 0{,}1956 & 9{,}40 \end{vmatrix} = 1{,}2 + 0{,}0349 = 1{,}2349;$$

$$y_1 = 1{,}7-\frac{1}{97{,}910}\begin{vmatrix} 8{,}64 & -0{,}434 \\ 4{,}91 & 0{,}1956 \end{vmatrix} = 1{,}7 - 0{,}0390 = 1{,}6610.$$

Продовжуючи процес обчислення при x_1 і y_1, отримаємо $x_2 = 1{,}2343$; $y_2 = 1{,}6615$ і т. д. до досягнення бажаної точності.

Приклад 3.7. Методом Ньютона наближено знайти додатний розв'язок системи

$$\begin{cases} x^2+y^2+z^2=1, \\ 2x^2+y^2-4z=0, \\ 3x^2-4y+z^2=0. \end{cases}$$

Розв'язання

Позначимо $\overline{f}(X)=\begin{bmatrix} x^2+y^2+z^2-1 \\ 2x^2+y^2-4z \\ 3x^2-4y+z^2 \end{bmatrix}$. Вибравши за початкове наближення $X^0=(0{,}5;0{,}5;0{,}5)$, отримаємо

$$\overline{f}(X^{(0)})=\begin{bmatrix} 0{,}25+0{,}25+0{,}25-1 \\ 0{,}5+0{,}25-2{,}00 \\ 0{,}75-2{,}00+0{,}25 \end{bmatrix}=\begin{bmatrix} -0{,}25 \\ -1{,}25 \\ -1{,}00 \end{bmatrix}.$$

Побудуємо матрицю Якобі

$F(X)=\begin{bmatrix}2x & 2y & 2z\\ 4x & 2y & -4\\ 6x & -4 & 2z\end{bmatrix}$, дістанемо $F(X^{(0)})=\begin{bmatrix}1 & 1 & 1\\ 2 & 1 & -4\\ 3 & -4 & 1\end{bmatrix}$.

Отримаємо систему рівнянь

$$\begin{bmatrix}1 & 1 & 1\\ 2 & 1 & -4\\ 3 & -4 & 1\end{bmatrix}\begin{bmatrix}\varepsilon_x^{(0)}\\ \varepsilon_y^{(0)}\\ \varepsilon_z^{(0)}\end{bmatrix}=\begin{bmatrix}0{,}25\\ 1{,}25\\ 1{,}00\end{bmatrix}.$$

Її розв'язок за методом Гаусса є $\varepsilon_x^{(0)}=0{,}375;\ \varepsilon_y^{(0)}=0;\ \varepsilon_z^{(0)}=-0{,}125.$

За знайденим приростом дістанемо перше наближення $X^{(1)}=X^{(0)}+\varepsilon^{(0)}$:

$$X^{(1)}=\begin{bmatrix}0{,}875\\ 1{,}5\\ 1{,}375\end{bmatrix}.$$

Далі обчислюємо друге наближення $X^{(2)}$. Маємо

$$\overline{f(X^{(1)})}=\begin{bmatrix}0{,}15625\\ 0{,}28125\\ 0{,}43750\end{bmatrix};\ F(X^{(1)})=\begin{bmatrix}1{,}75 & 1 & 0{,}75\\ 3{,}5 & 1 & -4\\ 5{,}25 & -4 & 0{,}75\end{bmatrix}.$$

Розв'язуючи систему рівнянь $F(X^{(1)})\overline{\varepsilon^{(1)}}=-\overline{f(X^{(1)})}$, отримуємо

$\varepsilon_x^{(1)}=-0{,}085183$; $\varepsilon_y^{(1)}=-0{,}0033784$; $\varepsilon_z^{(1)}=-0{,}0050675.$

Використовуючи знайдений приріст, будуємо друге наближення:

$$X^{(2)}=X^{(1)}+\overline{\varepsilon^{(1)}};\ X^{(2)}=\begin{bmatrix}0{,}78982\\ 0{,}49662\\ 0{,}36993\end{bmatrix}.$$

Аналогічно знаходимо третє наближення :

$$X^{(3)}=\begin{bmatrix}0{,}78521\\ 0{,}49662\\ 0{,}36992\end{bmatrix};\quad \overline{f(X^{(3)})}=\begin{bmatrix}0{,}00001\\ 0{,}00004\\ 0{,}00005\end{bmatrix}.$$

Його можна вважати шуканим розв'язком з точністю $\varepsilon=0{,}0001$.

Приклади до розділу 4 (частина 1)
Методи розв'язання диференціальних рівнянь

Пример 4.1. За методом Ейлера скласти таблицю рішення на відрізку [0;1] для рівняння $y'=y-\frac{2x}{y}$ з початковою умовою y(0) = 1, вибравши крок h = 0,2.

Розв'язання

Результати обчислень помістимо у таблицю, що заповнюється таким способом:

i	x_i	y_i	Δy_i	Точне $y=\sqrt{2x+1}$
0	0	1,0000	0,2000	1,0000
1	0,2	1,2000	0,1733	1,1832
2	0,4	1,3733	0,1561	1,3416
3	0,6	1,5294	0,1492	1,4832
4	0,8	1,6786	0,1451	1,6124
5	1,0	1,8237		1,7320

У першому рядку при $i = 0$ записується $x_0 = 0$, $y_0 = 1{,}000$ і за ними обчислюється $f(x_0, y_0) = 1$, а потім $\Delta y_0 = hf(x_0, y_0) = 0{,}2$. Тоді за формулою Ейлера одержуємо $y_1 = 1 + 0{,}2 = 1{,}2$.

Значення $x_1 = 0{,}2$ і $y_1 = 1{,}2000$ записуються у другому рядку при $i = 1$. Використовуючи їх, можна обчислити

$$f(x_1,y_1) = 0{,}8667;\ \Delta y_1 = hf(x_1,y_1) = 0{,}2\cdot 0{,}8667 = 0{,}1733.$$

Тоді $y_2 = y_1 + \Delta y_1 = 1{,}2 + 0{,}1733 = 1{,}3733$.

При $i = 2, 3, 4, 5$ обчислення ведуться аналогічно. В останньому стовпці таблиці для порівняння поміщені значення точного рішення.

З таблиці видно, що абсолютна похибка для y_5 дорівнює $\varepsilon = 1{,}8237 - 1{,}7320 = 0{,}0917$, що становить 5%.

Приклад 4.2. Застосовуючи метод Ейлера, скласти на відрізку [1;1,5] таблицю значень розв'язання рівняння

$$y'' + \frac{y'}{x} + y = 0$$

з початковими умовами $y(1) = 0{,}77$ і $y'(1) = -0{,}44$, вибравши крок h=0,1.

Розв'язання

Замінимо рівняння за допомогою підстановки $y' = z$, $y'' = z'$ системою рівнянь першого порядку

$$\begin{cases} y' = z; \\ z' = -\dfrac{z}{x} - y \end{cases}$$

з початковими умовами $y(1) = 0{,}77$ і $z(1) = -0{,}44$. Таким чином, маємо

$$\begin{cases} f_1(x,y,z) = z; \\ f_2(x,y,z) = -\dfrac{z}{x} - y. \end{cases}$$

Результати обчислення за формулою Ейлера записані в таблиці

i	x_i	y_i	ΔY_i	$f_{1i} = z_i$	Δz_i	$f_{2i} = -\dfrac{z_i}{x_i} - z_i$
0	1,0	0,77	–0,044	–0,44	–0,033	–0,33
1	1,1	0,726	–0,047	–0,473	–0,030	–0,296
2	1,2	0,679	–0,050	–0,503	–0,026	–0,260
3	1,3	0,629	–0,053	–0,529	–0,022	–0,222
4	1,4	0,576	–0,055	–0,551		
5	1,5	0,521				

Таблиця заповнюється таким чином. Записуємо в першому рядку $i = 0$, x_0=1,0; $y_0 = 0,77$; $z_0 = -0,44$.

Далі знаходимо

$$f_{10} = f_1(x_0, y_0, z_0) = z_0 = -0,44;$$

$$f_{20} = f_2(x_0, y_0, z_0) = -\frac{z_0}{x_0} - y_0 = -0,33.$$

Використовуючи формули Ейлера одержуємо

$\Delta y_0 = hf_{10} = 0,1 \cdot (-0,44) = -0,044;$ $\quad y_1 = y_0 + \Delta y_0 = 0,726;$

$\Delta z_0 = hf_{20} = 0,1 \cdot (-0,33) = -0,033;$ $\quad z_1 = z_0 + \Delta z_0 = -0,473.$

Таким чином, у другому рядку таблиці ми можемо записати $i = 1$; $x_1 = 1,1$; $y_1 = 0,726$; $z_1 = -0,473$. За цими значеннями знаходимо

$$f_{11} = f_1(x_1, y_1, z_1) = z_1 = -0,473;$$

$$f_{21} = f_2(x_1, y_1, z_1) = -\frac{z_1}{x_1} - y_1 = -0,296.$$

Відповідно

$\Delta y_1 = hf_{11} = 0,1 \cdot (-0,47) = -0,047;$ $\quad y_2 = y_1 + \Delta y_1 = 0,679;$

$\Delta z_1 = hf_{21} = 0,1 \cdot (-0,30) = -0,030;$ $\quad z_2 = z_1 + \Delta z_1 = -0,503.$

Заповнення таблиці при i =2, 3, 4, 5 здійснюються аналогічно.

Приклади до розділу 6 (частина 1)
Методи обробки експериментальних даних

Приклад 6.1. Знайти значення заданої таблично функції y = f(x) при x=0,4

i	0	1	2	3
x_i	0	0,1	0,3	0,5
y_i	–0,5	0	0,2	1

Розв'язання

Квадратична інтерполяція

$$y = a_i x^2 + b_i x + c_i.$$

Вибираємо три найближчі точки до $x_t = 0{,}4$

$x_{i-1} = 0{,}1;$ $x_i = 0{,}3;$ $x_{i+1} = 0{,}5.$

$y_{i-1} = 0;$ $y_i = 0{,}2;$ $y_{i+1} = 1.$

$$\left.\begin{array}{l} a_i x_{i-1}^2 + b_i x_{i-1} + c_i = y_{i-1} \\ a_i x_i^2 + b_i x_i + c_i = y_i \\ a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} 0{,}01a_i + 0{,}1b_i + c_i = 0; \\ 0{,}09a_i + 0{,}3b_i + c_i = 0{,}2; \\ 0{,}25a_i + 0{,}5b_i + c_i = 1; \end{array}\right.$$

$$A = \begin{vmatrix} 0{,}01 & 0{,}1 & 1 \\ 0{,}09 & 0{,}3 & 1 \\ 0{,}25 & 0{,}5 & 1 \end{vmatrix}; \quad \overline{B} = \begin{vmatrix} 0 \\ 0{,}2 \\ 1 \end{vmatrix}; \quad \overline{X} = \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \mathrm{A}^{-1}\overline{B}.$$

Знайдемо $A^{-1} = \begin{vmatrix} \frac{75}{6} & -\frac{75}{3} & \frac{25}{2} \\ -10 & \frac{45}{3} & -5 \\ \frac{15}{8} & -\frac{10}{8} & \frac{3}{8} \end{vmatrix};$

$$\overline{X} = \begin{vmatrix} a \\ b \\ c \end{vmatrix} = \begin{vmatrix} \frac{75}{6} & -\frac{75}{3} & \frac{25}{2} \\ -10 & \frac{45}{3} & -5 \\ \frac{15}{8} & -\frac{10}{8} & \frac{3}{8} \end{vmatrix} \cdot \begin{vmatrix} 0 \\ 0{,}2 \\ 1 \end{vmatrix};$$

$a = 0 - \frac{75}{3} * \frac{1}{5} + \frac{25}{2} = 7{,}5;$ $b = -2;$ $c = 0{,}125;$

$y = 7{,}5x^2 - 2x + 0{,}125;$ при $x = 0{,}4;$ $y = 0{,}525.$

Інтерполяційний многочлен Лагранжа

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

$$L(x) = \sum_{i=0}^{3} y_i \prod_{\substack{j=0 \\ j \neq i}}^{3} \frac{x - x_j}{x_i - x_j} = y_0 \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} + y_1 \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} +$$

$$+ y_2 \frac{(x-x_1)(x-x_0)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} =$$

$$= \frac{125}{3}x^3 - 30x^2 + \frac{91}{12}x - 0{,}5;$$

при $x = 0{,}4;$ $y \approx L(x) = 0{,}3999.$

Інтерполяційний многочлен Ньютона

Маємо випадок нерівновіддалених вузлів, $n = 3$;

Знаходимо поділені різниці:

$$f(x_0,x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{0-(-0,5)}{0,1} = 5;$$

$$f(x_1,x_2) = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{0,2-0}{0,3-0,1} = 1;$$

$$f(x_2,x_3) = \frac{f(x_3) - f(x_2)}{x_3 - x_2} = \frac{1-0,2}{0,5-0,3} = 4;$$

$$f(x_0,x_1,x_2) = \frac{f(x_1,x_2) - f(x_0,x_1)}{x_2 - x_0} = \frac{1-5}{0,3-0} = -\frac{40}{3};$$

$$f(x_1,x_2,x_3) = \frac{f(x_2,x_3) - f(x_1,x_2)}{x_3 - x_1} = \frac{4-1}{0,5-0,1} = \frac{15}{2};$$

$$f(x_0,x_1,x_2,x_3) = \frac{f(x_1,x_2,x_3) - f(x_0,x_1,x_2)}{x_3 - x_0} = \frac{\frac{15}{2} + \frac{40}{3}}{0,5} = \frac{125}{3}.$$

Результати розрахунків помістимо в таблицю:

n	x_n	f_n	$f(x_n, x_{n+1})$	$f(x_n, x_{n+1}, x_{n+2})$	$f(x_n, x_{n+1}, x_{n+2}, x_{n+3})$
0	0	–0,5			
1	0,1	0	5	–40/3	125/3
2	0,3	0,2	1	15/2	
3	0,5	1	4		

Використовуючи перші в стовпцях поділені різниці, одержимо

$$N_3(x) = -0,5 + (x-0)\cdot 5 + (x-0)(x-0,1)(-\frac{40}{3}) + (x-0)(x-0,1)\times$$

$$\times(x-0,3)\frac{125}{3} = \frac{125}{3}x^3 - 30x^2 + \frac{91}{12}x - 0,5.$$

Нагадаємо, що розрахунки інтерполяційного многочлена Ньютона виконуються за формулою

$$N_{n-1}(x_T) = y_1 + \sum_{k=1}^{n-1}(x_T - x_1)(x_T - x_2)\ldots(x_T - x_k)\Delta_1^k,$$

де x_T – поточна точка, у якій потрібно обчислити значення многочлена; Δ_1^k – поділені різниці порядку k, які обчислюються за такими формулами:

$$\Delta_i^1 = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}, \quad i = 1,\ldots,(n-1);$$

$$\Delta_i^2 = \frac{\Delta_i^1 - \Delta_{i+1}^1}{x_i - x_{i+2}}, \quad i = 1,\ldots,(n-2);$$

$$\Delta_i^k = \frac{\Delta_i^{k-1} - \Delta_{i+1}^{k-1}}{x_i - x_{i+k}}, \; i = 1,\ldots,(n-k).$$

Приклад 6.2. Розглянемо фрагмент таблиці функції $y = x + \sin x$

x_i	1,4	1,5	1,7	1,8
y_i	2,38545	2,49749	2,69166	2,77385

Розв'язання

Запишемо многочлен Лагранжа, використовуючи всю наявну інформацію, тобто, покладаючи $n = 3$, у вигляді

$$L_3(x) = 2{,}38548 \cdot \frac{(x-1{,}5)\cdot(x-1{,}7)\cdot(x-1{,}8)}{(1{,}4-1{,}5)\cdot(1{,}4-1{,}7)\cdot(1{,}4-1{,}8)} +$$
$$+2{,}49749 \cdot \frac{(x-1{,}4)\cdot(x-1{,}7)\cdot(x-1{,}8)}{(1{,}5-1{,}4)\cdot(1{,}5-1{,}7)\cdot(1{,}5-1{,}8)} +$$
$$+2{,}69166 \cdot \frac{(x-1{,}4)\cdot(x-1{,}5)\cdot(x-1{,}8)}{(1{,}7-1{,}4)\cdot(1{,}7-1{,}5)\cdot(1{,}7-1{,}8)} +$$
$$+2{,}77385 \cdot \frac{(x-1{,}4)\cdot(x-1{,}5)\cdot(x-1{,}7)}{(1{,}8-1{,}4)\cdot(1{,}8-1{,}5)\cdot(1{,}8-1{,}7)}.$$

Обчислимо значення $L_3(x)$ в точці $x = 1{,}6$, оцінивши спочатку $R_n(x)$ відповідно до формули $|R_n(x)| \le \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|$.

$\omega_4(1{,}6) = 0{,}2 \cdot 0{,}1 \cdot (-0{,}1) \cdot (-0{,}2) = 0{,}0004$,

$y^{(4)}(x) = \sin x$; $|\sin x| \le 1$.

Отже

$$|R_3(1{,}6)| \le \frac{1}{4!} \cdot 4 \cdot 10^{-4} \le 0{,}2 \cdot 10^{-4}.$$

Скористаємось формою запису інтерполяційного многочлена $L_n(x) = \omega_{n+1}(x)\sum_{i=0}^{n} \frac{Y_i}{x - x_i}$. Усі обчислення розташуємо в таблиці

i	$x-x_i$	$x_i-x_j,\ i\neq j$			$\prod_{i\neq j}(x_i-x_j)$,	$f(x_i)$	$\frac{Y_i}{(x-x_i)}$
0	0,2	– 0,1	– 0,3	– 0,4	– 0,012	2,38545	– 993,938
1	0,1	0,1	– 0,2	– 0,3	0,006	2,49749	4162,48
2	– 0,1	0,3	0,2	– 0,1	– 0,006	2,69166	4486,10
3	– 0,2	0,4	0,3	0,1	0,012	2,77385	– 1155,77

$$L_3(1,6)=\omega_4(1,6)\cdot\sum_{i=0}^{3}\frac{Y_i}{(1,6-x_i)},$$

$$L_3(1,6)=2,59955.$$

Для порівняння наведемо значення функції $y=x+\sin x$ для $x=1,6$ з п'ятьма точними десятковими знаками: $y(1,6)=2,59957$.

Приклад 6.3. За допомогою формули Сімпсона обчислити $I=\int_{\pi/4}^{\pi/2}\frac{\sin x}{x}dx$ з точністю $\varepsilon = 10^{-3}$.

Розв'язання

Виберемо крок h.

$$R_c=-\frac{h^4(b-a)}{180}f^{IV}(\xi);\quad \xi\in[a,b],\ \text{тобто}\ \xi\in[\pi/4,\pi/2];$$

$$\frac{h^4(b-a)}{180}\max_{[a,b]}\left|f^{IV}(x)\right| < 0,5\cdot10^{-3}.$$

Обчислимо $f^{IV}(x)$

$$f^{IV}(x)=\frac{\sin x}{x}+4\frac{\cos x}{x^2}-12\frac{\sin x}{x^3}-24\frac{\cos x}{x^4}+24\frac{\sin x}{x^5}.$$

Оцінимо $|f^{IV}|$ на відрізку $[\pi/4, \pi/2]$. Скористаємося величинами $\frac{\sin x}{x}\left(1-\frac{12}{x^2}+\frac{24}{x^4}\right)$ і $\frac{4\cos x}{x^2}\left(\frac{6}{x^2}-1\right)$. Вони позитивні та спадають, отже, їх максимальне значення в точці $x = \pi/4$.

При цьому $\left|f^{(IV)}(x)\right|\le\frac{\sin x}{x}\left(1-\frac{12}{x^2}+\frac{24}{x^4}\right)+\frac{4\cos x}{x^2}\left(\frac{6}{x^2}-1\right)< 81$. Таким чином, $R\le\frac{h^4\cdot\pi/4}{180}\cdot81<0,5\cdot10^{-3}$; $h^4<14\cdot10^{-4}$; $h\le0,19$.

З іншого боку, для даного методу h вибирається з урахуванням того, щоб відрізок $[\pi/4, \pi/2]$ ділився на парне число відрізків. Цим двом вимогам відповідає $h=\pi/24 = 0,13 < 0,19$, при якому $n=\frac{b-a}{h}=6$. Тоді, щоб похибка округлення не перевищила $0,5\cdot10^{-3}$, досить обчислення виконати з 4-ма знаками після коми.

Складемо таблицю $y = \dfrac{\sin x}{x}$ з $h = \pi/24 = 7° 30' = 0{,}1309$

i	x_i^0	x_i	$\sin x$	y_0, y_6	y_{2m}	y_{2m-1}
0	45° 00′	0,7854	0,7071	0,9003		
1	52° 30′	0,9163	0,7934			0,8659
2	60° 00′	1,0472	0,8660		0,8270	
3	67° 30′	1,1781	0,9239			0,7843
4	75° 00′	1,3090	0,9659		0,7379	
5	82° 30′	1,4399	0,9914			0,6885
6	90° 00′	1,5708	1,0000	0,6366		
Сума				1,5369	1,5649	2,3386

Для $n = 6$ за формулою Сімпсона

$$\int_{\pi/4}^{\pi/2} \frac{\sin x}{x} = \frac{h}{3}[(y_0 + y_6) + 4(y_1 + y_3 + y_5) + 2(y_2 + y_4)] = 0{,}6118 \approx 0{,}612.$$

Прикла 6.4. За формулою Гаусса при $n = 5$ обчислити $I = \int_0^1 \dfrac{dx}{1+x^2}$.

Розв'язання

Зробимо заміну змінної $x = 1/2 + t \cdot 1/2$, тоді

$$I = 2\int_{-1}^{1} \frac{dt}{4 + (t+1)^2}.$$

Складемо таблицю значень підінтегральної функції.

i	ξ_i	$f(\xi_i)$	q_i
1	–0,9061179846	0,24945107	0,236926885
2	–0,538469310	0,23735995	0,478628670
3	0	0,2	0,568888889
4	0,538469310	0,15706211	0,478628670
5	0,906179846	0,13100114	0,236926885

За формулою Гаусса $\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} q_i f(\xi_i) + R$ визначимо

$I = 2[q_1 f(\xi_1) + q_2 f(\xi_2) + ... + q_3 f(\xi_3)] = 0{,}78539816$;

Приклад 6.5. Розглянемо приклад розробки математичної моделі за результатами активного експерименту. Розглядається експеримент з трьома вихідними y_1, y_2, y_3 та трьома вхідними x_1, x_2, x_3 змінними. Вихідні дані для проведення серії експериментів:

Параметр		x_1	x_2	x_3
рівень:	середній	15	50	75
	верхній (1)	20	60	100
	нижній (-1)	10	40	50
інтервал		5	10	25

Результати реалізації активного ПФЕ:

№ досліду	Вхідні параметри			Вихідні параметри		
	x_1	x_2	x_3	y_1	y_2	y_3
1	1	1	1	1.5	92	6.4
2	-1	1	1	0.5	163	5.5
3	1	-1	1	1	110	4.8
4	-1	-1	1	0.7	142	3.4
5	1	1	-1	1.2	77	6.7
6	-1	1	-1	0.3	152	5.8
7	1	-1	-1	0.4	105	5.1
8	-1	-1	-1	0.3	143	4.0

Після проведення розрахунків за формулами

$$b_0 = \frac{1}{N}\sum_{j=1}^{N} y_j x_{j0},$$

$$b_i = \frac{1}{N}\sum_{j=1}^{N} y_j x_{ji},$$

$$b_{ik} = \frac{1}{N}\sum_{j=1}^{N} y_j x_{ji} x_{jk}$$

отримані такі коефіцієнти регресії:

	b_0	b_1	b_2	b_3	b_{12}	b_{13}	b_{23}
y_1	0,73	0,28	0,0145	0,019	0,017	0,004	-0,0004
y_2	123	-27	-0,193	0,375	-0,9	0,125	0,0250
y_3	5,19	0,54	0,085	-0,019	-0,008	0,003	0,0003

Перевірка адекватності моделі: припустимо, що при проведенні

паралельних експериментів для кожної комбінації параметрів при середньоквадратичних відхиленнях $\sigma_{y_1} = 0{,}086$, $\sigma_{y_2} = 4{,}5$, $\sigma_{y_3} = 0{,}2$ середньоквадратичні відхилення коефіцієнтів регресії становлять, відповідно, $\sigma_{b_1}^{(y_1)} = 0{,}003$; $\sigma_{b_2}^{(y_2)} = 0{,}16$; $\sigma_{b_3}^{(y_3)} = 0{,}006$. Тоді $\Delta b^{(y_i)} = t_{\alpha,N} * \sigma_b^{(y_i)}$, де $t_{\alpha,N}$ – критерій Стьюдента, що визначається за статистичними таблицями і залежить від рівня значущості та кількості експериментів N. Тоді для $t_{0,05;8} = 2{,}3$ можна отримати інтервали довіри відповідно для коефіцієнтів рівняння

$$\Delta b^{(y_1)} = 0{,}0069;\ \Delta b^{(y_2)} = 0{,}368;\ \Delta b^{(y_3)} = 0{,}014.$$

Якщо вилучити для кожного з рівнянь коефіцієнти, що менші за ці інтервали, отримаємо остаточні рівняння:

$$y_1 = 0{,}73 + 0{,}28x_1 + 0{,}0145x_2 + 0{,}019x_3 + 0{,}019x_1x_2;$$

$$y_2 = 123 - 27x_1 + 0{,}375x_3 - 0{,}9x_1x_2;$$

$$y_3 = 5{,}19 + 0{,}54x_1 + 0{,}085x_2 - 0{,}019x_3.$$

ДОДАТОК Б

ПРИКЛАДИ ВИКОРИСТАННЯ ПАКЕТА MATHCAD ДЛЯ РОЗВ'ЯЗАННЯ ЧИСЕЛЬНИХ ЗАДАЧ

Пакет прикладних програм MathCad

MathCad – програмний засіб, середовище для виконання на комп'ютері різноманітних математичних і технічних розрахунків, що надають користувачу інструменти для роботи з формулами, числами, графіками й текстами, оснащене простим в освоєнні графічним інтерфейсом.

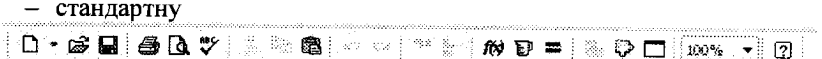
MathCad – система візуальних математичних розрахунків. Основна ідея MathCad полягає в тому, що обчислювані вирази записуються у візуальній формі, максимально наближеній до математичного запису, звичного для людини. Звичайно, MathCad не єдиний з пакетів свого класу. Достатньо згадати також універсальні математичні пакети Maple, MathLab, Mathematica. Але особливістю саме пакета MathCad є те, що він відповідає принципу *WYSIWYG (What You See Is What You Get* – «що бачите, те й одержуєте»).

Можливості системи

1. Числові розрахунки зі скалярами, матрицями та векторами. Можливі розрахунки з використанням комплексних чисел.
2. Аналітичні перетворення: інтегрування, диференціювання, обчислення меж, сум і добутків рядів, спрощення, перетворення Лапласа і Фур'є та інше.
3. Визначення законів обчислення елементів матриць, що дозволяє реалізувати ітераційні обчислення, у тому числі за рекурентними формулами.
4. Робота з функціями: інтерполяція, екстраполяція, чисельне інтегрування, матричні функції та інше.
6. Побудова двовимірних і тривимірних графіків різних виглядів.
7. Розв'язання систем лінійних і нелінійних рівнянь.
8. Розв'язання оптимізаційних завдань виду: знайти значення змінних, при яких функція приймає мінімальне або максимальне значення.
9. Розв'язання диференціальних рівнянь (звичайні диференціальні рівняння й системи рівнянь; рівняння Пуассона й Лапласа).
10. Елементи програмування.

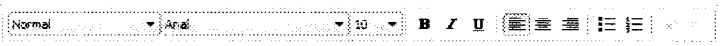
Основне вікно програми містить три панелі:

– стандартну



,

– форматування тексту



– математичну

а також робочу область із автоматично створеною сторінкою обчислень. На математичній панелі перебувають кнопки, при натисканні на які відкриваються додаткові панелі з шаблонами введення різних виразів: *Calculator* (знаки деяких основних функцій і операцій), *Calculus* (шаблони операцій інтегрування, диференціювання, меж та інших), *Evaluation* (оператори присвоювання та обчислення), *Graph* (графіки), *Greek* (символи грецького алфавіту), *Matrix* (операції векторного й скалярного добутків, транспонування, векторної суми, обчислення визначника матриці), *Programming* (елементи програмування), *Boolean* (логічні операції), *Symbolic* (різного роду аналітичні перетворення).

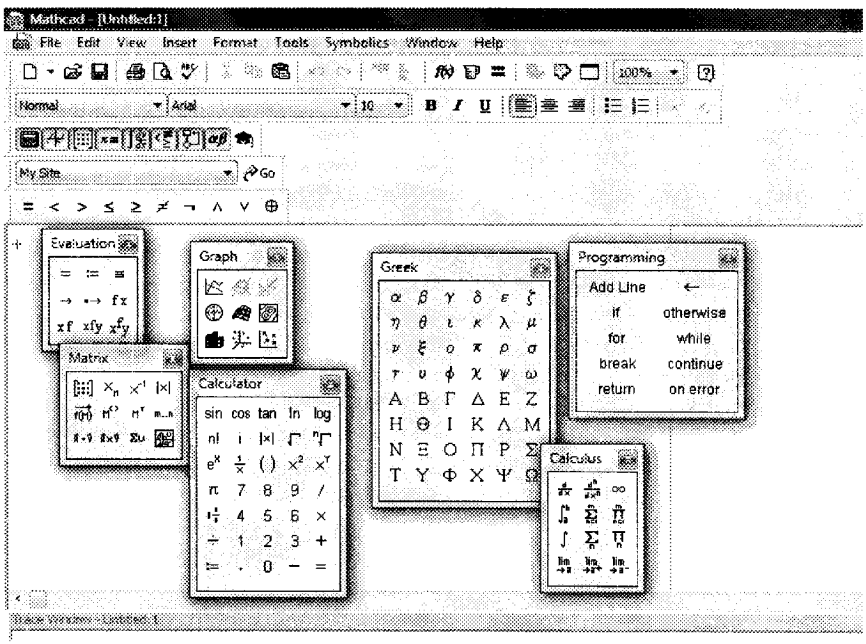


Рисунок Б.1 – Вікно MathCad13

Для визначення будь-якої змінної потрібно набрати її ім'я і знак «:», MathCad додасть знак «=», після чого можна ввести її значення або вираз. Для змінної з індексом (масив, вектор, матриця) після імені потрібно набрати квадратну дужку і вказати індекси через кому. Змінні можна використовувати після їхнього визначення правіше та нижче.

Будь-яка формула може бути записана в будь-якому місці аркуша. Необхідно клацнути мишею в передбачуваній точці введення формули і там повинен з'явитися покажчик уведення. Введення звичайно здійснюється як із клавіатури, так і за допомогою миші. Під час уведення або редагування формули MathCad обводить її рамкою

$$\frac{a+b}{2}+x$$

Приклади простих обчислень:

$$\int_a^b \exp(x)\,dx \rightarrow \exp(b) - \exp(a)$$

$$\frac{5-\sqrt{3}}{2} = 1.634$$

$$\frac{1+2j}{1-2j} = -0.6 + 0.8i$$

Рисунок Б.2 – Фрагмент сторінки MathCad з обчисленими виразами

На рис. Б.2 наведені приклади аналітичного та числового обчислень. Ліва частина, враховуючи знаки обчислення, вводиться користувачем, права частина – результат, обчислений системою MathCad. Крім інтегралів система дозволяє аналітично обчислювати вирази, що містять похідні різних порядків, суми та добутки рядів, односторонні та двосторонні границі. У розділі *Symbolics* верхнього меню також доступно багато операцій аналітичного перетворення виразів (спрощення, згортання, розкладання в ряд, перетворення Лапласа і Фур'є та ін.)

Для того, щоб на екрані відображались результати обчислень з необхідною кількістю знаків, встановимо формат чисел, викликавши меню **Format → Result**.... В діалоговому вікні **Result Format** (рис. Б.3) на вкладці **Number Format**, у групі **General** змінимо значення **Number of decimal places** з 3 на 6. Ту ж цифру поставимо в значення **Exponential threshold**. Клацнемо на кнопці **OK** діалогового вікна. Тепер ми зможемо бачити на екрані результати обчислень з шістьома знаками після коми.

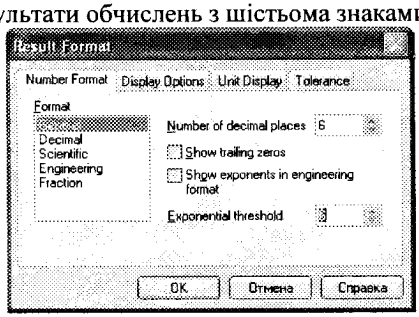
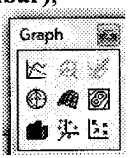


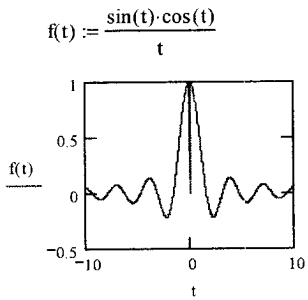
Рисунок Б.3 – Діалогове вікно Result Format

Побудова графіка функції. Розглянемо найпростіший спосіб побудови графіка функції. Для цього виконаємо такі дії:

1. Визначити функцію. Наприклад, $f(t) = \frac{\sin(t) \cdot \cos(t)}{t}$.
2. Ввести шаблон графіка в Декартовій системі координат за допомогою меню **Insert →Graph→X-Y-Plot**, або за допомогою панелі графіків (**Graph Toolbar**);



3. З'явиться незаповнений шаблон. Шаблон являє собою великий пустий прямокутник з місцями введення даних у вигляді маленьких чорних прямокутників (маркери введення), які розміщені біля осей майбутнього графіка. Введемо ім'я змінної t в середнє поле введення біля осі абсцис і ім'я функції $f(t)$ в середнє поле введення біля осі ординат;
4. Клацніть мишею поза областю графіка – він буде побудований.



MathCad дозволяє будувати такі типи графіків (рис. Б.4):

– графіки функцій однієї змінної f(x);
– графіки параметрично заданих функцій у полярній системі координат;
– графіки поверхні для функцій двох змінних z(x,y);
– контурні графіки для функцій двох змінних z(x,y);
– стовпчикові графіки для функцій двох змінних z(x,y);
– крапкові графіки поверхні для функцій двох змінних z(x,y);
– графіки векторного поля (у вигляді стрілок).

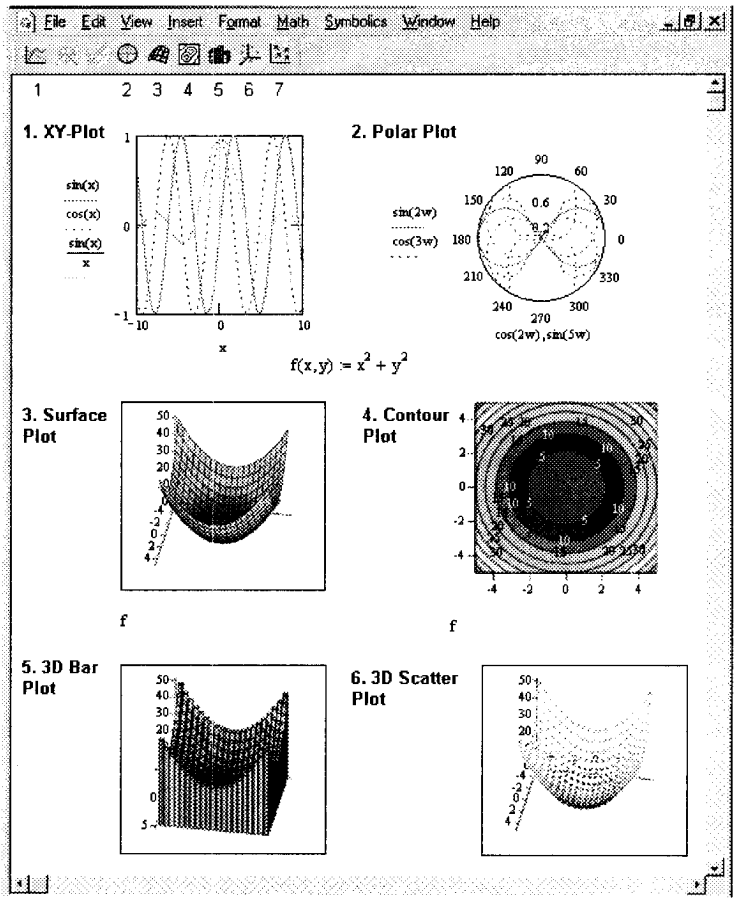


Рисунок Б.4 – Основні види графіків MathCad

Базові операції MathCad

Приклад 1. Виконати такі операції для $i := 1 .. 10$

$$\sum_i i = 55$$

$$\prod_i (i + 1) = 3.992 \times 10^7$$

$$\int_{0}^{0.4} x^2 \cdot \log(x+2)\,dx = 7.711\times 10^{-3}$$

$$\int_{0.8}^{1.2} \frac{\cot(2\cdot x)}{\sin(2\cdot x)^2}\,dx = -0.298$$

$x := 2$

$$\frac{d}{dx}x^5 = 80$$

$$\frac{d}{dx}\sin(x) = -0.416$$

Приклад 2. Визначити змінні *a:=3.4, b:=6.22, c*≡0.149 (причому змінну *c* – глобально) та вираз:

$a := 3.4$

$b := 6.22$

$c \equiv 0.149$

$$Z := \frac{2\cdot a\cdot b + \sqrt[3]{c}}{\sqrt{\left(a^2 + b^{a+c}\right)\cdot c}}$$

$$N := e^{\sin(c)}\cdot\cos\left(\frac{a}{b}\right)$$

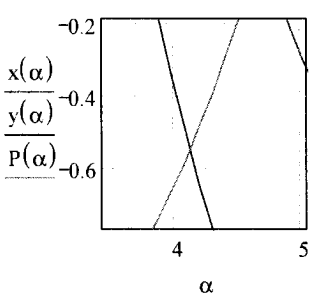
$Z = 4.2928313$

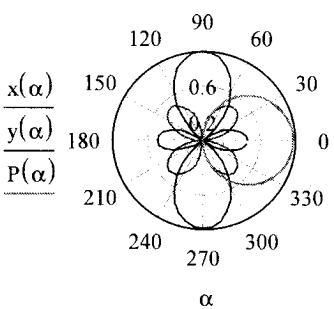
$N = 0.991$

Приклад 3. Побудувати декартові (**X-Y Plot**) і полярні (**Polar Plot**) графіки нижченаведених функцій:

$x(\alpha) := \cos(\alpha)\cdot\sin(\alpha)$ $\quad y(\alpha) := 1.5\cos(\alpha)^2 - 1$

$P(\alpha) := \cos(\alpha)$ $\quad \alpha := 0, 0.105 .. 2\pi$





Xper := 2.205 $\qquad \alpha := \frac{\pi}{2}$

Yper := −0.47335

$x(\alpha) = 0 \quad y(\alpha) = -1$

Розв'язання задач лінійної алгебри в середовищі MathCad.

До основних функцій лінійної алгебри відносять:

1. Додавання матриць;
2. Віднімання матриць;
3. Множення матриці на скаляр;
4. Множення матриць;
5. Транспонування матриці: $[A(n,m)]^T = B(m,n)$,
   $a_{ij} = b_{ji}$ де $i = 1, ..., n$, $j = 1, ..., m$;
6. Знаходження оберненої матриці.

Нехай задана матриця $A = \begin{vmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{vmatrix}$

1. Створення транспонованої матриці в середовищі MathCad:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \qquad A^T = \begin{pmatrix} 2 & 3 & 4 \\ 3 & 2 & 3 \\ 1 & 2 & 1 \end{pmatrix}$$

2. Знаходження оберненої матриці:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \qquad A^{-1} = \begin{pmatrix} -0.5 & 0 & 0.5 \\ 0.625 & -0.25 & -0.125 \\ 0.125 & 0.75 & -0.625 \end{pmatrix}$$

$$A := \begin{pmatrix} 2 & 3 & 3 \\ 1 & 2 & 3 \\ 2 & 4 & 5 \end{pmatrix} \qquad A^{-1} = \begin{pmatrix} 2 & 3 & -3 \\ -1 & -4 & 3 \\ 0 & 2 & -1 \end{pmatrix}$$

3. Додавання матриць:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \qquad B := \begin{pmatrix} 2 & 3 & 3 \\ 1 & 2 & 3 \\ 2 & 4 & 5 \end{pmatrix} \qquad A + B = \begin{pmatrix} 4 & 6 & 4 \\ 4 & 4 & 5 \\ 6 & 7 & 6 \end{pmatrix}$$

4. Віднімання матриць

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \quad B := \begin{pmatrix} 2 & 3 & 3 \\ 1 & 2 & 3 \\ 2 & 4 & 5 \end{pmatrix} \quad A - B = \begin{pmatrix} 0 & 0 & -2 \\ 2 & 0 & -1 \\ 2 & -1 & -4 \end{pmatrix}$$

5. Множення матриць:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \quad B := \begin{pmatrix} 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{pmatrix}$$

$$A \cdot B = \begin{pmatrix} 23 & 6 \\ 27 & 7 \\ 29 & 8 \end{pmatrix}$$

$B \cdot A =$ ∎∎

The number of rows and/or columns in these arrays do not match.

6. Множення матриці на число:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \quad C := 2 \quad A \cdot C = \begin{pmatrix} 4 & 6 & 2 \\ 6 & 4 & 4 \\ 8 & 6 & 2 \end{pmatrix}$$

7. Знаходження детермінанта:

$$A := \begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 1 \end{pmatrix} \quad |A| = 8$$

Розв'язання нелінійних рівнянь
Знаходження коренів нелінійних рівнянь в середовищі MathCad

Для найпростіших рівнянь виду *f(x)=0* розв'язання в MathCad відбувається за допомогою функції *root*:

root(f(x1, x2, ...), x1, a, b)

Повертає значення *x1*, що належить відрізку *[a, b]*, при якому вираз або функція *f(x)* перетворюється в 0. Обидва аргументи цієї функції повинні бути скалярами. Функція повертає скаляр.

f(x1, x2, ...) – функція, визначена у робочому документі, або вираз. Вираз повинен повертати скалярні значення.

x1 – ім'я змінної, яка використовується у виразі. Цій змінній перед використанням функції *root* необхідно присвоїти числове значення.

MathCad використовує його як початкове наближення для пошуку кореня.

Для знаходження коренів виразу, що має вигляд

$$v_n x^n + ... + v_2 x^2 + v_1 x + a_0,$$

краще використовувати функцію *polyroots*, ніж *root*. На відміну від функції *root,* функція *polyroots* не вимагає початкового наближення і повертає відразу всі корені, як дійсні, так і комплексні.

Polyroots(v) – повертає корінь полінома степеня *n*. Коефіцієнти полінома знаходяться у векторі *v* довжиною *n+1*. Повертає вектор довжини *n*, що складається з коренів полінома.

v – вектор, що містить коефіцієнти полінома. Вектор *v* зручно створювати, використовуючи команду **Symbolics => Polynomial coefficients.**

Приклад 4. Знайти розв'язок нелінійного рівняння $f(x) = x^2 - 2 = 0$. Початкове наближення $x_0 = -1$.

1) Задаємо коефіцієнти рівняння:

a2 := 1

a1 := 0

a0 := –2

2) Задаємо вигляд функції:

$F(x) := a2 \cdot x^2 + a1 \cdot x + a0$

3) Задаємо початкове наближення:

x := –1

4) Знаходимо перший корінь:

x1 := root(F(x), x) x1 = –1.414

5) Знаходимо інші корені:

$x2 := \text{root}\left[\frac{F(x)}{(x - x1)}, x\right]$ x2 = 1.414

Відповідь: $x_1 = -1.414, x_2 = 1.414$.

Приклад 5. Знайти розв'язок нелінійного рівняння $f(x) = x + \tan(0.2x)$ методом Ньютона. Початкове наближення $x_0 = 7$.

Задамо початкові умови та точність обчислення:

n := 100 i := 1..n $\varepsilon := 10^{-4}$

$df(x) := \frac{d}{dx} f(x)$

f(x) := x + tan(0.2x)

$x_0 := 7$

Обчислення першого наближення за формулою Ньютона:

$$x_1 := x_0 - \frac{f(x_0)}{df(x_0)}$$

Реалізація ітераційного процесу за методом Ньютона з використанням функції *until*:

$$x_{i+1} := until\left(\left|x_i - x_{i-1}\right| - \varepsilon, x_i - \frac{f(x_i)}{df(x_i)}\right)$$

Визначення числа ітерацій, за які ітераційний процес зійшовся

$j := last(x)$

$$x = \begin{pmatrix} 7 \\ 5.385 \\ 1.553 \\ 0.018 \\ 2.468\times 10^{-8} \\ 0 \end{pmatrix}$$

$j = 5$

$x_{j-1} = 2.468\times 10^{-8}$

Відповідь: $x = 2.468\times 10^{-8}$

Приклад 6. Знайти розв'язок нелінійного рівняння $f(x) = x^3 - 3x^2 + x - 2 = 0$. Початкове наближення $x_0 = -1$.

$a3 := 1$

$a2 := -3$

$a1 := 1$

$a0 := -2$

$F(x) := a3\cdot x^3 + a2\cdot x^2 + a1\cdot x + a0$

$x := -1$

$x1 := root(F(x), x)$ $\quad x1 = 2.893$

$i := \sqrt{-1}$ $\quad x := 1 + i\cdot 1$

$$x2 := root\left(\frac{F(x)}{x - x1}, x\right) \quad x2 = 0.053 + 0.83i$$

$$x3 := root\left[\frac{F(x)}{(x - x1)(x - x2)}, x\right] \quad x3 = 0.053 - 0.83i$$

Відповідь: $x_1 = 2.983$, $x_2 = 0.053 + 0.83i$, $x_3 = 0.054 - 0.829i$

Коментар. Даний приклад ілюструє, що другий та третій корені даним

способом знаходяться з певною похибкою.

Приклад 7. Знайти розв'язок рівняння $x^4 + 9x^3 + 31x^2 + 59x + 60$.

У випадку полінома можна застосувати функцію *polyroots*:

$$c := \begin{pmatrix} 60 \\ 59 \\ 31 \\ 9 \\ 1 \end{pmatrix} \qquad \text{polyroots}(c) = \begin{pmatrix} -4 \\ -3 \\ -1 - 2i \\ -1 + 2i \end{pmatrix}$$

Іншим способом отримати розв'язок є символьне розв'язання за допомогою команди *Symbolic=>Solve for variable*

$$x^4 + 9x^3 + 31x^2 + 59x + 60 \qquad \begin{pmatrix} -4 \\ -3 \\ -1 + 2i \\ -1 - 2i \end{pmatrix}$$

Розв'язання систем лінійних рівнянь в середовищі MathCad

Систему лінійних рівнянь в середовищі MathCad можна розв'язати декількома способами. Найбільш простий з них з використанням функції *lsolve(M, v),* де вектор рішення *x* такий, що $Mx=v$.

Приклад 8. Нехай задана система рівнянь:

$$\begin{cases} x_1 + 2 \cdot x_2 + 3 \cdot x_3 = 1; \\ x_1 + 3 \cdot x_2 + 2 \cdot x_3 = 2; \\ 2 \cdot x_1 + 3 \cdot x_2 + 4 \cdot x_3 = 4. \end{cases}$$

Для розв'язання рівняння в даному середовищі необхідно подати його в матричній формі (матриця лівої частини системи та матриця вільних членів):

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 3 & 4 \end{pmatrix} \qquad B := \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

Пошук розв'язку системи:

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 3 & 4 \end{pmatrix} \qquad B := \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

$$x := \text{lsolve}(A, B) \qquad x = \begin{pmatrix} 4 \\ 0 \\ -1 \end{pmatrix}$$

Перевірка знайденого розв'язку

$$A \cdot x - B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Відповідь: $x_1 = 4$; $x_2 = 0$; $x_3 = -1$.

Іншим способом є розв'язання системи рівнянь за допомогою функцій *Given* та *Find*. Для розв'язання системи рівнянь необхідно виконати такі дії.

– Задати початкове наближення для всіх невідомих, що входять у систему рівнянь. MathCad розв'язує систему за допомогою ітераційних методів.

– Надрукувати ключове слово *Given*. Воно вказує MathCad, що далі буде введено систему рівнянь.

– Ввести рівняння та нерівності в будь-якому порядку. Використовувати знак = з панелі *Boolean* або [Ctrl] += .

– Ввести функцію *Find*(*z*1, *z*2, . . .), яка повертає точний розв'язок системи рівнянь. Число аргументів повинно бути рівним числу невідомих.

Приклад 9. Початкові наближення:

x1 := 0
x2 := 0
x3 := 0
x4 := 0
x5 := 0
Given
x1 + x2 + x3 + x4 + x5 = 15
x1 – x3 + 7x4 = 26
x1 – 2x2 + 3x3 – 4x4 = –10
x1 – x2 + 2x3 + 3x5 = 20
x1 + x3 – x4 + 10x5 = 50

$$\text{Find}(x1, x2, x3, x4, x5) = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Розглянемо розв'язання системи лінійних рівнянь **методом ітерацій** в середовищі MathCad:

У MathCad існують спеціальні функції для обчислення норм матриць:

normi(A) – повертає невизначену норму матриці A;

norm1(A) – повертає L1, норму матриці A;

norme(A) – повертає Евклідову норму матриці A.

Приклад 10. Розв'яжемо нижченаведену систему:

$$\begin{cases} 100x_1 + 6x_2 - 2x_3 = 200 \\ 6x_1 + 200x_2 - 10x_3 = 600 \\ x_1 - 2x_2 - 100x_3 = 500 \end{cases}.$$

Приведемо дану систему до вигляду:

$$\begin{cases} x_1 = 2 - 0{,}06x_2 + 0{,}02x_3 \\ x_2 = 3 - 0{,}03x_1 + 0{,}05x_3 \\ x_3 = 5 - 0{,}01x_1 - 0{,}02x_2 \end{cases}.$$

В матричній формі її можна записати так:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} + \begin{pmatrix} 0 & -0{,}06 & 0{,}02 \\ -0{,}03 & 0 & 0{,}05 \\ -0{,}01 & 0{,}02 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\beta := \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} \qquad \alpha := \begin{pmatrix} 0 & -0.06 & 0.02 \\ -0.03 & 0 & 0.05 \\ -0.01 & -0.02 & 0 \end{pmatrix}$$

Перевірка достатньої умови збіжності:

$$\text{normi}(\alpha) = 0.08$$

$$\text{norme}(\alpha) = 0.089$$

$$\text{norm1}(\alpha) = 0.08$$

$x^{\langle 0 \rangle} := \beta$ – визначення початкового наближення розв'язку;

$i := 0..4$ – визначення кількості ітерацій;

$x^{\langle i+1\rangle} := \beta + \alpha \cdot x^{\langle i\rangle}$ – формула обчислення за методом ітерацій.

Матриця наближених розв'язків:

$$x = \begin{pmatrix} 2 & 1.92 & 1.907 & 1.907 & 1.907 & 1.907 \\ 3 & 3.19 & 3.188 & 3.189 & 3.189 & 3.189 \\ 5 & 4.92 & 4.917 & 4.917 & 4.917 & 4.917 \end{pmatrix}$$

Оцінка похибки:

$$\varepsilon := \frac{\left|x^{\langle 5\rangle} - x^{\langle 4\rangle}\right|}{\left|x^{\langle 4\rangle}\right|} \qquad \varepsilon = 8.612 \times 10^{-8}$$

Розглянемо розв'язання системи лінійних рівнянь **методом Гаусса** у середовищі MathCad з використанням таких функцій:

rref(A) – повертається ступінчаста форма матриці A;

augment(A, У) – повертається масив, сформований розташуванням A і B пліч-о-пліч. Масиви A і B повинні мати однакове число рядків;

submatrix(A, ir, jr, ic, jc) – повертається субматриця, що складається з усіх елементів з *ir* по *jr* і стовпцях з *ic* по *jc*.

Приклад 11. Розв'язати систему рівнянь методом Гаусса.

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 15 \\ x_1 - x_3 + 7x_4 = 26 \\ x_1 - 2x_2 + 3x_3 - 4x_4 = -10 \\ x_1 - x_2 + 2x_3 + 3x_5 = 20 \\ x_1 + x_3 - x_4 + 10x_5 = 50 \end{cases}$$

$$A := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 7 & 0 \\ 1 & -2 & 3 & -4 & 0 \\ 1 & -1 & 2 & 0 & 3 \\ 1 & 0 & 1 & -1 & 10 \end{pmatrix} \qquad B := \begin{pmatrix} 15 \\ 26 \\ -10 \\ 20 \\ 50 \end{pmatrix}$$

C := augment(A, B)

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 15 \\ 1 & 0 & -1 & 7 & 0 & 26 \\ 1 & -2 & 3 & -4 & 0 & -10 \\ 1 & -1 & 2 & 0 & 3 & 20 \\ 1 & 0 & 1 & -1 & 10 & 50 \end{pmatrix}$$

G := rref(C)

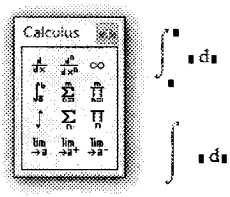
$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 & 5 \end{pmatrix}$$

$z := \text{submatrix}(G, 0, 4, 5, 5)$

$$z = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Розв'язання задач чисельного інтегрування в середовищі MathCad

Щоб обчислити визначений інтеграл в середовищі MathCad, потрібно надрукувати його оператор за допомогою панелі *Calculus* натисканням кнопки зі значком інтеграла. З'явиться символ інтеграла з декількома комірками, у які потрібно ввести нижню і верхню межі інтегрування, підінтегральну функцію та змінну інтегрування.



Інтеграли в середовищі MathCad розв'язуються так
Варіант А:

$$\int_a^b 3x^2 + 2x + 1\,dx \rightarrow b^3 + b^2 + b - a^3 - a^2 - a$$

Варіант Б:

$a := 1 \qquad b := 10$

$$\int_a^b 3x^2 + 2x + 1\,dx \rightarrow 1107$$

Варіант В:

$a := 1 \qquad b := 10$

$$\left(\int_a^b 3x^2 + 2x + 1\, dx \right) = 1.107 \times 10^3$$

Приклад 12. Обчислити інтеграл $\int_1^{10} \sin(x)(x+1)dx$ за формулами трапецій, Сімпсона та Гаусса.

Введемо початкові умови (число ітерацій, межі інтегрування, крок):

$n := 8 \qquad i := 0..n \qquad a := 1 \qquad b := 10$

$$h := \frac{b-a}{n}$$

$f(x) := \sin(x) \cdot (x+1)$

$x_i := a + h \cdot i \qquad y_i := f(x_i)$

Обчислимо точне значення даного інтеграла:

$$\int_1^{10} \sin(x)(x+1)\, dx = 8.925$$

Метод трапецій:

$$Itr := \frac{b-a}{n} \cdot \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i \right)$$

$Itr = 7.658$

Значення похибки за методом трапеції:

$x := 1$

$$Rtr := \frac{n \cdot h^3}{12} \cdot \frac{d^2}{dx^2} f(x) \qquad |Rtr| = 0.572$$

Метод Сімпсона:

$j := 1..4$

$k := 1..3$

$$Isim := \frac{h}{3} \left[y_0 + y_8 + 4 \cdot \left(\sum_j y_{2 \cdot j - 1} \right) + 2 \left(\sum_k y_{2k} \right) \right]$$

$Isim = 9.081$

Значення похибки за методом Сімпсона:

$x := 1$

$$Rsim := \frac{(b-a)\cdot h^4}{80}\cdot\frac{d^4}{dx^4}f(x)$$

$|Rsim| = 0.086$

Метод Гаусса:

$A_i :=$

0.10122
0.22238
0.3137
0.36268
0.36268
0.3137
0.22238
0.10122

$t_i :=$

−0.96028
−0.79666
−0.52553
−0.18343
0.18343
0.52553
0.79666
0.96028

$$x_i := \frac{a+b}{2} + t_i\cdot\left(\frac{b-a}{2}\right)$$

$$Igaus := \frac{b-a}{2}\cdot\left[\sum_{i=1}^{8}\left(A_i\cdot f(x_i)\right)\right]$$

$Igaus = 8.925$

Як видно з результатів обчислення, найбільш точне значення отримано за допомогою методу Гаусса.

Приклад 13. Обчислити інтеграл $\int_0^{3.2}\sqrt{(x^4-x^3+8)}dx$ за методом Монте-Карло при n=10.

Використана у прикладі функція *runif* повертає вектор з m випадкових значень, що мають універсальний розподіл на інтервалі [a;b].

a := 0 b := 3.2 n := 10

$A := runif(10, 0, 3.2)$ $A =$

	0
0	4.059·10-3
1	0.619
2	1.872
3	1.121
4	2.633
5	0.557
6	2.274
7	0.973
8	0.293
9	0.471

$I0 := \sqrt{(A_0)^4 - (A_0)^3 + 8}$ $I0 = 2.828$

$I1 := \sqrt{(A_1)^4 - (A_1)^3 + 8}$ $I1 = 2.812$

$I2 := \sqrt{(A_2)^4 - (A_2)^3 + 8}$ $I2 = 3.704$

$I3 := \sqrt{(A_3)^4 - (A_3)^3 + 8}$ $I3 = 2.858$

$I4 := \sqrt{(A_4)^4 - (A_4)^3 + 8}$ $I4 = 6.149$

$I5 := \sqrt{(A_5)^4 - (A_5)^3 + 8}$ $I5 = 2.815$

$I6 := \sqrt{(A_6)^4 - (A_6)^3 + 8}$ $I6 = 4.792$

$I7 := \sqrt{(A_7)^4 - (A_7)^3 + 8}$ $I7 = 2.824$

$I8 := \sqrt{(A_8)^4 - (A_8)^3 + 8}$ $I8 = 2.825$

$I9 := \sqrt{(A_9)^4 - (A_9)^3 + 8}$ $I9 = 2.819$

$Summa := I0 + I1 + I2 + I3 + I4 + I5 + I6 + I7 + I8 + I9$ $Summa = 34.428$

$I := \frac{b - a}{n} \cdot Summa$ $I = 11.017$

Чисельне розв'язання диференціальних рівнянь у пакеті MathCad

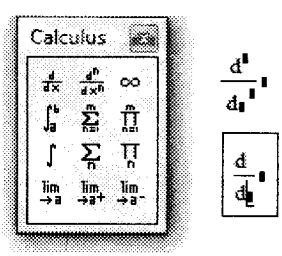
В MathCad можна обчислювати похідні скалярних функцій будь-якої кількості аргументів, від 0-го до 5-го порядку включно. І функції, і аргументи можуть бути як дійсними, так і комплексними.

Для того, щоб продиференціювати функцію f(x) у деякій точці потрібно:

– визначити точку x, у якій буде обчислена похідна, наприклад x:=1;

– ввести оператор диференціювання натисканням кнопки *Derivative* на панелі *Calculus*;

– у комірках, що з'явилися, ввести функцію f(x) і ім'я аргументу x;



Наприклад,

$x := 0.01$

$\frac{d}{dx}(\cos(x) \cdot \ln(x)) = 100.041$

Для розв'язання диференціальних рівнянь MathCad надає користувачеві бібліотеку вбудованих функцій **Differential Equation Solving**, призначених для чисельного розв'язання диференціальних рівнянь.

Вбудована функція *odesolve* призначена для розв'язання диференціальних рівнянь лінійних щодо старшої похідної. Функція *odesolve* вирішує поставлене завдання методом Рунге-Кутта з фіксованим кроком. Звертання до функції має вигляд

$$Y:=odesolve(x,b,step) \text{ або } Y:=odesolve(x,b),$$

де Y – ім'я функції, що містить значення знайденого розв'язку, x – змінна інтегрування, b – кінець проміжку інтегрування, *step* – крок, що використовується при інтегруванні рівняння методом Рунге-Кутта.

Приклад 14. Знайти за допомогою функції *odesolve* на відрізку $[0,4\pi]$ рішення задачі Коші.

$$y'' - y' \sin x + y = \frac{x}{2\pi}, \quad y(0) = 0, \quad y'(0) = 1$$

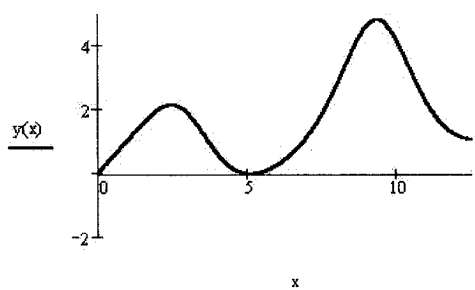
Given

$y''(x) - \sin(x)\cdot y'(x) + y(x) = \frac{x}{2\pi}$

$y(0) = 0$

$y'(0) = 1$

$y:=odesolve(x, 4\pi)$



У результаті змінній y привласнюються значення чисельного розв'язку задачі Коші на відрізку $[0,\ 4\pi]$.

В MathCad розв'язати задачу Коші для диференціальних рівнянь, а також для систем диференціальних рівнянь можна за допомогою таких функцій:

– *rkfixed(y, x1, x2, npoints, D)* – розв'язання задачі на відрізку методом Рунге-Кутта з постійним кроком;

– *Rkadapt(y, x1, x2, npoints, D)* – розв'язання задачі на відрізку методом Рунге-Кутта з автоматичним вибором кроку;

– *rkadapt(y, x1, x2, acc, npoints, D, kmax, save)* – розв'язання задачі в заданій точці методом Рунге-Кутта з автоматичним вибором кроку;

– *Bulstoer(y, x1, x2, npoints, D)* – розв'язання задачі на відрізку методом Буліpша-Штера;

– *bulstoer(y, x1, x2, acc, npoints, D, kmax, save)* – розв'язання задачі в заданій точці методом Буліpша-Штера;

– *stiffr(y, x1, x2, acc, D, J, kmax, save)* – розв'язання задачі для твердих систем на відрізку з використанням алгоритму Розенброка.

Приклад 15. Знайти на відрізку $[0,\pi]$ наближений розв'язок рівняння $y' = \sin(xy)$, що задовольняє початкову умову $y(0)=1$, і побудувати графік знайденого розв'язку.

Вирішимо завдання використовуючи алгоритм Рунге-Кутта з фіксованим кроком на сітці з 20 рівновіддалених вузлів.

Перш ніж вводити диференціальне рівняння, визначимо номер першого компонента вектора 1 (а не нулем, як передбачається за замовчуванням).

ORIGIN := 1

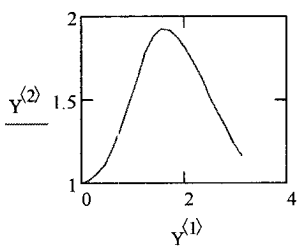
$y_1 := 1$

$D(x,y) := \sin(x \cdot y_1)$

$Y := \text{rkfixed}(y, 0, \pi, 20, D)$

$Y =$

	1	2
1	0	1
2	0.157	1.012
3	0.314	1.05
4	0.471	1.115
5	0.628	1.208
6	0.785	1.331
7	0.942	1.477
8	1.1	1.633
9	1.257	1.774
10	1.414	1.874
11	1.571	1.921
12	1.728	1.916
13	1.885	1.872
14	2.042	1.801
15	2.199	1.714
16	2.356	1.619



Приклад 16. Знайти на відрізку [0,3] наближений розв'язок задачі Коші

$$\begin{cases} y_1' = -y_2 + \sin(xy_3) \\ y_2' = -y_1^2 \\ y_3' = -y_3 - y_1 \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \\ y_3(0) = 1 \end{cases}$$

і побудувати графік знайденого розв'язку.

Вирішимо завдання, використовуючи алгоритм Рунге-Кутта з фіксованим кроком на сітці з 20 рівновіддалених вузлів.

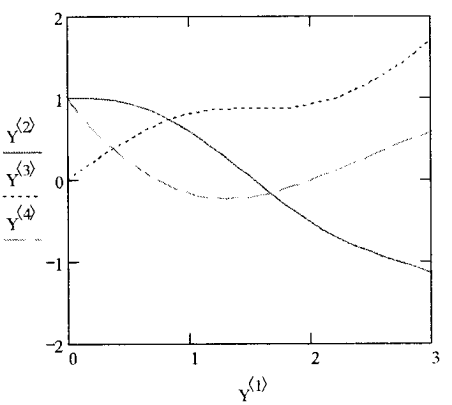
ORIGIN := 1

$$y := \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \qquad D(x,y) := \begin{bmatrix} -y_2 + \sin(x \cdot y_3) \\ (y_1)^2 \\ -y_3 - y_1 \end{bmatrix}$$

Y := rkfixed(y, 0, 3, 30, D)

Y =

	1	2	3	4
1	0	1	0	1
2	0.1	0.999	0.1	0.81
3	0.2	0.995	0.199	0.638
4	0.3	0.984	0.298	0.483
5	0.4	0.964	0.393	0.344
6	0.5	0.932	0.483	0.221
7	0.6	0.889	0.566	0.113
8	0.7	0.833	0.64	0.021
9	0.8	0.764	0.704	-0.057
10	0.9	0.683	0.757	-0.121
11	1	0.591	0.797	-0.17
12	1.1	0.49	0.827	-0.205
13	1.2	0.382	0.846	-0.227
14	1.3	0.268	0.857	-0.236
15	1.4	0.151	0.861	-0.234
16	1.5	0.032	0.862	-0.22



Приклад 17. Знайти розв'язок диференціального рівняння $y'(x) = 2x^2 + 2y$ $(y(0) = 1;\ \ h = 0{,}2)$.

$f(X,Y) := 2X^2 + 2Y$

$n := 10 \qquad i := 0..n$

$h := 0.2 \qquad x_0 := 0 \qquad y_0 := 1$

$x_i := x_0 + i \cdot h$

Метод Ейлера:

$y_{i+1} := y_i + h \cdot f(x_i, y_i)$

Метод Рунге-Кутта:

$Yrk_0 := y_0$

$k1(f,X,Y,h) := h \cdot f(X,Y)$

$k2(f,X,Y,h) := h \cdot f\left(X + \frac{h}{2}, Y + \frac{k1(f,X,Y,h)}{2}\right)$

$k3(f,X,Y,h) := h \cdot f\left(X + \frac{h}{2}, Y + \frac{k2(f,X,Y,h)}{2}\right)$

$k4(f,X,Y,h) := h \cdot f(X + h, Y + k3(f,X,Y,h))$

$RK(f,X,Y,h) := \frac{1}{6}(k1(f,X,Y,h) + 2k2(f,X,Y,h) + 2k3(f,X,Y,h) + k4(f,X,Y,h))$

$Yrk_{i+1} := Yrk_i + RK(f, x_i, Yrk_i, h)$

Точне значення заданого диференціального рівняння:

$t(x) := 1.5e^{2x} - x^2 - x - 0.5$

Отримані значення:

Метод Ейлера $y_i =$	Метод Рунге-Кутта $Yrk_i =$	Точне значення $t(x_i) =$
1	1	1
1.4	1.498	1.498
1.976	2.278	2.278
2.83	3.52	3.52
4.107	5.488	5.49
6.005	8.581	8.584
8.807	13.39	13.395
12.906	20.798	20.807
18.853	32.123	32.139
27.418	49.331	49.357
39.681	75.353	75.397

Обчислимо похибки отриманих результатів за допомогою методів Ейлера та Рунге-Кутта:

$$\frac{|t(x_{10}) - y_{10}|}{t(x_{10})} \cdot 100 = 47.371$$

$$\frac{|t(x_{10}) - Yrk_{10}|}{t(x_{10})} \cdot 100 = 0.059$$

Знайдемо розв'язок рівняння за допомогою вбудованої функції **rkfixed**:

$Y_0 := 1$

$D(X, Y) := 2X^2 + 2Y_0$

$Z := \text{rkfixed}(Y, 0, 1, 10, D)$ $\quad Z^{\langle 1 \rangle} =$

	0
0	1
1	1.222
2	1.498
3	1.843
4	2.278
5	2.827
6	3.52
7	4.393
8	5.489
9	6.864
10	8.583

Проаналізувавши результати, можна зробити висновок, що найвищу точність обчислень можна отримати за допомогою методу Рунге-Кутта.

Розв'язання задач інтерполювання в середовищі MathCad

Пакет MathCad дозволяє виконувати інтерполяцію двох типів: кусково-лінійну і сплайнову. При лінійній інтерполяції MathCad з'єднує існуючі точки даних прямими лініями. Це виконується функцією **linterp**, яка повертає значення функції при її лінійному інтерполюванні.

linterp(vx, vy, x) використовує вектори даних vx і vy, щоб повернути інтерпольоване значення y, яке відповідає третьому аргументу x. Аргументи vx й vy повинні бути векторами однакової довжини. Вектор vx повинен містити дійсні значення, розташовані в порядку зростання.

Ця функція з'єднує точки даних відрізками прямих, створюючи таким чином ламану. Значення, що інтерполюється, для конкретного x є ордината y відповідної точки ламаної.

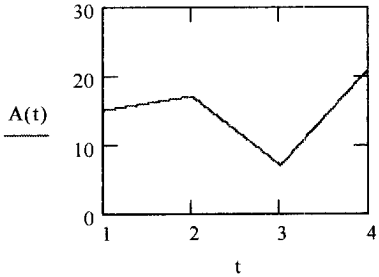
Приклад 18. Побудувати інтерполяційний многочлен Лагранжа третього степеня для функції f, заданої таблично та знайти наближене значення функції в точці $x = 2{,}5$.

x	1	2	3	4
y	15	17	7	21

$x := (1 \;\; 2 \;\; 3 \;\; 4)^T$ $\qquad y := (15 \;\; 17 \;\; 7 \;\; 21)^T$

$A(t) := \text{linterp}(x, y, t)$ $\quad A(2.5) = 12$

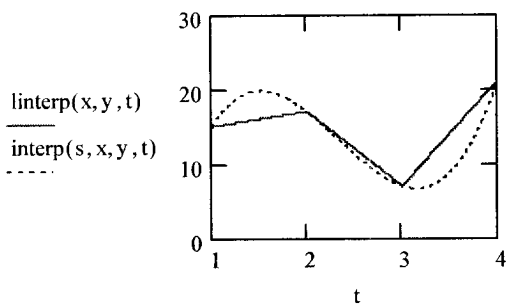
Графічне відображення інтерполяційного многочлена Лагранжа.



При невеликій кількості вузлових точок (менш ніж 10) лінійна інтерполяція є достатньо грубою. При її застосовуванні навіть перша похідна функції інтерполяції має різкі стрибки у вузлових точках.

В більшості практичних задач експериментальні дані бажано з'єднувати не ламаною лінією, а гладкою кривою, що дозволяє зробити сплайн-інтерполяція. При її застосуванні початкова функція замінюється відрізками кубічних поліномів, що проходять через три суміжні вузлові точки.

Порівняння результатів, отриманих в результаті застосовування лінійної інтерполяції і інтерполяції сплайнами:



Для здійснення сплайнової інтерполяції використовується функція **interp(vs,vx,vy,x)**.

interp(vs,vx,vy,t) – функція, що апроксимує дані векторів x і y кубічними сплайнами, де vs – вектор других похідних, що створюється за допомогою однією з функцій **cspline**, **pspline** или **lspline**; vx – вектор дійсних даних аргументу, елементи якого розташовані в порядку зростання; vy – вектор дійсних даних того ж розміру; x – значення аргументу, при якому обчислюється функція.

Іншими словами функція **interp(vs,vx,vy,t)** повертає значення y, що інтерполюється, яке відповідає аргументу x. Як вже зазначалося, вектор *vs* обчислюється на основі векторів даних vx і vy однією з функцій **pspline**, **lspline** або **cspline**. Вони повертають вектор коефіцієнтів похідних 2-го

порядку, що ми будемо називати vs. Аргументи vx й vy повинні бути дійсними векторами однакової довжини. Значення вектора vy повинні бути розташовані в порядку зростання.

Приклад 19. Використовуючи дані прикладу 9 провести сплайн-інтерполяцію.

$$x := (1 \;\; 2 \;\; 3 \;\; 4)^T \qquad y := (15 \;\; 17 \;\; 7 \;\; 21)^T$$

Інтерполяція сплайнами з використанням функції **lspline:**

$s1 := \text{lspline}(x, y)$

$A1(t) := \text{interp}(s1, x, y, t)$

$A1(2.5) = 11.1$

Інтерполяція сплайнами з використанням функції **pspline:**

$s2 := \text{pspline}(x, y)$

$A2(t) := \text{interp}(s2, x, y, t)$

$A2(2.5) = 11.25$

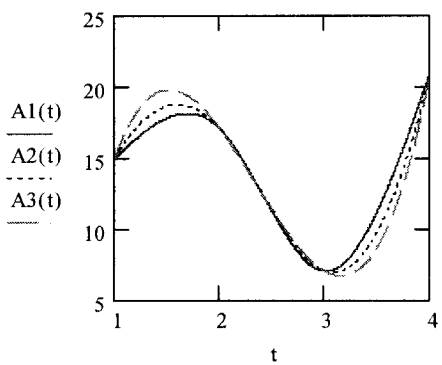
Інтерполяція сплайнами з використанням функції **cspline:**

$s3 := \text{cspline}(x, y)$

$A3(t) := \text{interp}(s3, x, y, t)$

$A3(2.5) = 11.25$

Результати інтерполяції заданих даних, отримані з використанням функцій **lspline**, **pspline**, **cspline**:



Запрограмуємо реалізацію методу Лагранжа. Формула Лагранжа в пакеті MathCad набуде такого вигляду:

$$f(x) := \sum_i yi_i \cdot \prod_j \text{if}\left(i = j, 1, \frac{x - xi_j}{xi_i - xi_j}\right)$$

В результаті отримаємо:

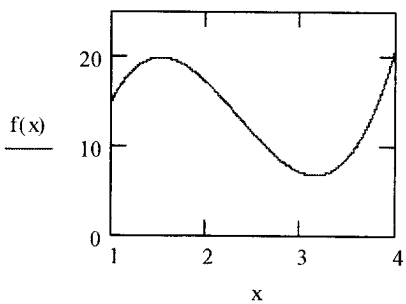
$$xi := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \qquad yi := \begin{pmatrix} 15 \\ 17 \\ 7 \\ 21 \end{pmatrix}$$

$n := \text{length}(xi) - 1 \qquad i := 0..n \qquad j := 0..n$

$$f(x) := \sum_i yi_i \cdot \prod_j \text{if}\left(i = j, 1, \frac{x - xi_j}{xi_i - xi_j}\right)$$

$f(2.5) = 11.25$

Результат інтерполяції, отриманий в результаті обчислень:



Приклад 20. Інтерполяція сплайнами:

ORIGIN := 1 $\qquad i := 1..4 \qquad j := 1..4$

$x_i :=$ $\qquad y_j :=$

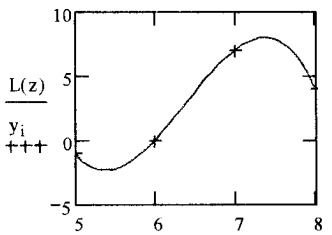
5
6
7
8

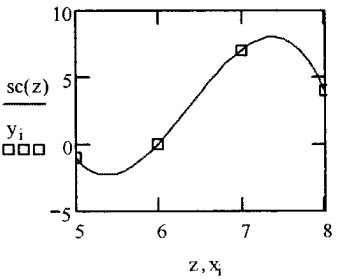
−1
0
7
4

$z := 5, 5.1..8$

$$L(z) := \sum_i \left(y_i \cdot \prod_j \text{if}\left(i = j, 1, \frac{z - x_j}{x_i - x_j} \right) \right) \qquad sc(z) := \text{interp}(\text{cspline}(x, y), x, y, z)$$

$L(5.75) = -1.438$

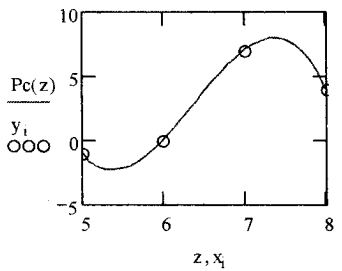




$$P(z) := -1 + z - 5 + 6 \cdot (z - 5) \cdot \frac{z - 6}{2} - \frac{16 \cdot (z - 5) \cdot (z - 6) \cdot (z - 7)}{6}$$

$$Pc(z) := 644 - \frac{952}{3} \cdot z + 51 \cdot z^2 - \frac{8}{3} \cdot z^3$$

$Pc(5.75) = -1.437$



Приклад 21. Апроксимація функції методом найменших квадратів

ORIGIN := 1 i := 1..8

$x_i :=$	$y_i :=$
1.6	3.06
2.6	3.33
3.6	3.52
4.6	3.68
5.6	3.81
6.6	3.91
7.6	4.01
8.6	4.11

a := slope(x, y) a = 0.143

$b := intercept(x, y)$ $b = 2.947$

$f(x, a1, b1) := a1 \cdot x + b1$

$s(a1, b1) := \sum_i \left(y_i - f(x_i, a1, b1)\right)^2$

$a1 := 1$ $b1 := 1$

Given

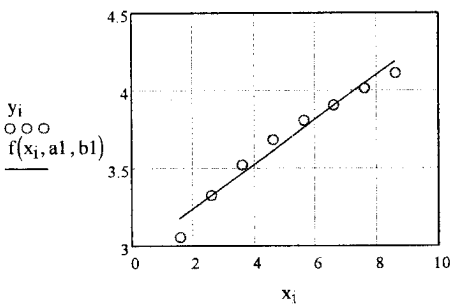
$s(a1, b1) = 0$

$I = 1$

$\begin{pmatrix} a1 \\ b1 \end{pmatrix} := Minerr(a1, b1)$

$a1 = 0.143$ $b1 = 2.947$

$s(a1, b1) = 0.032$



Елементи програмування в MathCad

Традиційне програмування, спрощений варіант якого застосований в MathCad і здійснюється за допомогою панелі інструментів Programming, має ряд істотних переваг:

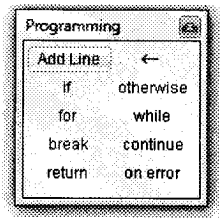
– можливість застосування циклів і умовних операторів;

– простота створення функцій і змінних, що вимагають декількох простих кроків;

– можливість створення функцій, що містять закритий для решти документа код, враховуючи переваги використання локальних змінних і обробку виняткових ситуацій.

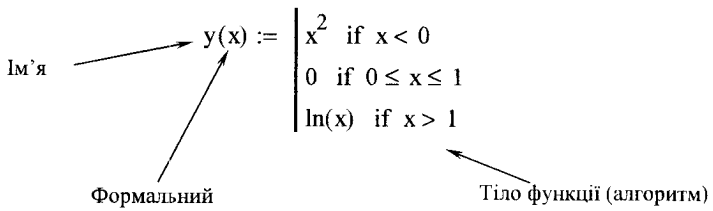
Щоб почати створення програмного модуля, слід натиснути на панелі **Programming** кнопку **Add Line**. Потім, якщо приблизно відомо, скільки

рядків кодів міститиме програма, можна створити потрібну кількість ліній повторними натисненнями кнопки **Add Line**.



Програмний модуль є функцією, описаною із застосуванням як сугубо алгоритмічних засобів (операторів), так і засобів вхідної мови MathCad. Як і в традиційному програмуванні, при визначенні функції вказується її ім'я, список формальних параметрів (аргументів) і алгоритм обчислення значення – тіло функції. Всі змінні, які вводяться всередині модуля, охоплюючи формальні параметри, є локальними стосовно всього документа.

Нижче наведено приклад програми функції:



Таблиця Б.1 – Огляд програмних операторів MathCad

Команда	Функція	Приклад
Add Line	Додати новий програмний рядок	
←	Присвоювання значення локальній змінній.	y ← 0
if	Умовний оператор (оператор розгалуження) if; умова повинна стояти після if, а оператор, що виконується, якщо виконано задану умову, – перед if.	f(x) := |-x if x < 0 x otherwise f(-3) = 3 f(4) = 4

otherwise	Оператор, що задає альтернативну гілку умовного оператора. Позначає оператор, що повинен бути виконаний, якщо умова оператора if не виконується.	$f(x) := \begin{cases} -x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$ $f(-3) = 3 \qquad f(4) = 4$
for	Оператор циклу з параметром. За ключовим словом for слідує змінна-лічильник, а після символа приналежності вводиться проміжок зміни цієї змінної.	$Sum(n) := \begin{cases} s \leftarrow 0 \\ \text{for } i \in 1..n \\ \quad s \leftarrow s + i \end{cases}$ $Sum(4) = 10$
while	Оператор циклу с передумовою. Внутрішні оператори циклу будуть виконуватися доти, доки буде істинною умова, що слідує за ключовим словом while. Приклад показує застосування циклу для знаходження нулів функції методом дотичних Ньютона.	$N(x, f, f_x) := \text{while } \lvert f(x) \rvert > 10^{-6}$ $x \leftarrow x - \frac{f(x)}{f_x(x)}$ $N(2, \sin, \cos) = 3.142$
break	Опертор дострокового припинення циклу або програми. Служить для передчасного завершення циклу, щоб, наприклад, уникнути зациклення або занадто тривалих обчислень.	break if $i \geq 10$
continue	Оператор переходу до наступної ітерації. Служить для передчасного завершення поточної ітерації циклу; сам цикл при цьому триває.	continue if $x \geq 10$
retutn	Передчасне завершення програми; зазначене в комірці значення буде повернуто.	return y
on error	Оператор, що визначає значення, яке повертається у випадку виникнення помилки. Якщо при обчисленні виразу expr2 виникла помилка, обчислюється вираз expr1.	expr1 on error expr2

Приклад 22. Розглянемо реалізацію методу трапецій для обчислення інтегралів за допомогою програмних операторів MathCad. Обчислити інтеграл $\int_0^{3.2} \sqrt{(x^4 - x^3 + 8)}\,dx$ при n=10.

a := 0 b := 3.2 n := 10

$$h := \frac{b - a}{n} \qquad h = 0.32$$

$$f(x) := \sqrt{x^4 - x^3 + 8}$$

$$I1 := \left(\frac{f(a) + f(b)}{2} \right)$$

trap(a, b, n, h, I1) :=
for i ∈ 1 .. n − 1
  x1 ← a + h
  I1 ← I1 + f(x1)
a ← x1
I1 ← I1
I1

trap(a, b, n, h, I1) = 31.309

Приклад 23. Розглянемо реалізацію методу Гаусса для розв'язання системи лінійних рівнянь за допомогою програмних операторів MathCad.

Функція **gauss** містить два параметри: a – матриця системи рівнянь і c – вектор вільних членів. Алгоритм функції містить два основних цикли за змінною i. Перший цикл реалізує прямий хід, а другий – зворотний хід методу. У тілі першого циклу передбачена процедура перестановки рівнянь у випадку, якщо провідний елемент виявляється рівним нулю. Як результат, функція повертає вектор x – вектор розв'язку системи.

```
gauss(a, c) := | n ← last(c)
               | for i ∈ 1 .. n − 1
               |   if a_{i,i} = 0
               |     t ← c_i
               |     c_i ← c_{i+1}
               |     c_{i+1} ← t
               |     for k ∈ 1 .. n
               |       t ← a_{i,k}
               |       a_{i,k} ← a_{i+1,k}
               |       a_{i+1,k} ← t
               |   for k ∈ (i + 1) .. n
               |     p ← a_{k,i} / a_{i,i}
               |     for j ∈ (i + 1) .. n
               |       a_{k,j} ← a_{k,j} − p · a_{i,j}
               |     c_k ← c_k − p · c_i
               | for i ∈ 1 .. n
               |   x_i ← 0
               | for i ∈ n, n − 1 .. 1
               |   s ← 0
               |   for j ∈ i .. n
               |     s ← s + a_{i,j} · x_j
               |   x_i ← (c_i − s) / a_{i,i}
               | x
```

Для ілюстрації роботи функції розв'яжемо таку систему:

$$\begin{cases} -7x_2 + 2x_3 = 7 \\ -3x_1 + 2x_2 + 6x_3 = 4 \\ 5x_1 - x_2 + x_3 = 6 \end{cases}$$

$$A := \begin{pmatrix} 0 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 0 \end{pmatrix} \qquad c := \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}$$

$$x := \text{gauss}(A, c) \qquad x = \begin{pmatrix} 1 \\ -1 \\ 1.5 \end{pmatrix}$$

Перевірка:

$$A \cdot x = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}$$

Приклад 24. Розглянемо реалізацію методу половинного ділення для знаходження коренів нелінійного рівняння за допомогою програмних операторів MathCad.

Створена функція **bs_root (bisection root)** містить чотири формальних параметри: f – функція, нулі якої шукаються; a і b – границі відрізка локалізації кореня; ε – абсолютна похибка. Основний алгоритм реалізований за допомогою циклічного оператора while, у тілі якого відбувається обчислення середньої точки відрізка, перевірка знаків функції на відрізку та вибір нового відрізка. Для підрахунку числа ітерацій використовується змінна it. Як результат, функція повертає наближене значення кореня та кількість ітерацій.

```
bs_root(f, a, b, ε) := | it ← 0
                       | while |b − a| > 2·ε
                       |     | it ← it + 1
                       |     | c ← (a + b)/2
                       |     | b ← c  if f(a)·f(c) < 0
                       |     | a ← c  otherwise
                       | ans_{0,0} ← c
                       | ans_{0,1} ← it
                       | ans
```

ДОДАТОК В

ПРИКЛАДИ ВИКОРИСТАННЯ ПАКЕТА MATLAB ДЛЯ РОЗВ'ЯЗАННЯ ЧИСЕЛЬНИХ ЗАДАЧ

Пакет прикладних програм MATLAB

MATLAB (скорочення від англ. «Matrix Laboratory») – термін, який стосується пакета прикладних програм для вирішення завдань технічних обчислень, а також використання в цьому пакеті мови програмування. MATLAB використовують понад 1 000 000 інженерних і наукових працівників, він працює на більшості сучасних операційних систем, в тому числі GNU / Linux, Mac OS, Solaris і Microsoft Windows.

MATLAB як мова програмування був розроблений Клівом Моулером (англ. Cleve Moler) наприкінці 1970-х років, коли він був деканом факультету комп'ютерних наук в Університеті Нью-Мексико. Метою розробки було дати студентам факультету можливість використання програмних бібліотек Linpack і EISPACK без необхідності вивчення Фортрану. Незабаром нова мова поширилась серед інших університетів і надзвичайно зацікавила вчених, що працюють в галузі прикладної математики. До цих пір в Інтернеті можна знайти версію 1982 р., написану на Фортрані, що розповсюджується з відкритим вихідним кодом. Інженер Джон Літтл (англ. John N. Little) познайомився з цією мовою під час візиту Кліва Моулера в Стенфордський університет в 1983 р. Зрозумівши, що нова мова має великий комерційний потенціал, він об'єднався з Клівом Моулером і Стівом Бангертом (англ. Steve Bangert). Спільними зусиллями вони переписали MATLAB на C і заснували в 1984 р. компанію The MathWorks для подальшого розвитку. Ці переписані на C бібліотеки довгий час були відомі під ім'ям JACKPAC. Спочатку MATLAB призначався для проектування систем управління (основна спеціальність Джона Літтла), але швидко завоював популярність у багатьох інших наукових та інженерних сферах. Він також широко використовувався і в освіті, зокрема, викладанні лінійної алгебри та чисельних методів.

Мова MATLAB є високорівневою інтерпретованою мовою програмування, що охоплює структури даних засновані на матрицях, широкий спектр функцій, інтегроване середовище розробки, об'єктно-орієнтовані можливості та інтерфейси до програм, написаних іншими мовами програмування.

Програми, написані на MATLAB, бувають двох типів – функції та скрипти. Функції мають вхідні та вихідні аргументи, а також власний робочий простір для зберігання проміжних результатів обчислень і змінних. Скрипти ж використовують загальний робочий простір. Як скрипти, так і функції не компілюються в машинний код і зберігаються у

вигляді текстових файлів. Існує також можливість зберігати так звані pre-parsed програми – функції та скрипти, оброблені у вигляд, зручний для машинного виконання. У загальному випадку такі програми виконуються швидше звичайних, особливо якщо функція містить команди побудови графіків.

Застосування

1. Математика та обчислення. MATLAB надає користувачеві велику кількість (кілька сотень) функцій для аналізу даних, що покривають практично всі області математики, зокрема:

– Матриці та лінійна алгебра – алгебра матриць, лінійні рівняння, власні значення і вектори, сингулярність, факторизація матриць та інші;

– Поліноми і інтерполяція – корені многочленів, операції над многочленами та їх диференціювання, інтерполяція та екстраполяція кривих та інші;

– Математична статистика та аналіз даних – статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур'є та інші;

– Обробка даних – набір спеціальних функцій, охоплюючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування (в квадратурах) та інші;

– Диференціальні рівняння – розв'язання диференціальних та диференціально-алгебраїчних рівнянь, диференціальних рівнянь з запізненням, рівнянь з обмеженнями, рівнянь в приватних похідних та інші;

– Розріджені матриці – спеціальний клас даних пакета MATLAB, що використовується в спеціалізованих програмах;

– Цілочисельна арифметика – виконання операцій цілочисельної арифметики в середовищі MATLAB.

2. *Розробка алгоритмів.* MATLAB надає зручні засоби для розробки алгоритмів, охоплюючи високорівневі з використанням концепцій об'єктно-орієнтованого програмування. У ньому є всі необхідні засоби інтегрованого середовища розробки, враховуючи відладчик і профайлер. Функції для роботи з цілими типами даних полегшують створення алгоритмів для мікроконтролерів і інших застосувань, де це необхідно.

3. *Візуалізація даних.* У складі пакета MATLAB є велика кількість функцій для побудови графіків, у тому числі тривимірних, візуального аналізу даних і створення анімаційних роликів. Вбудоване середовище розробки дозволяє створювати графічні інтерфейси користувача з різними елементами управління, такими як кнопки, перемикачі та інші. За допомогою компонента *MATLAB Compiler* ці графічні інтерфейси можуть бути перетворені в самостійні додатки, для запуску яких на інших комп'ютерах необхідна встановлена бібліотека *MATLAB Component Runtime.*

4. *Зовнішні інтерфейси.* Пакет MATLAB містить різні інтерфейси для отримання доступу до зовнішніх підпрограм, що написані іншими мовами програмування, даних, клієнтів і серверів, спілкується через технології *Component Object Model (COM)* або *Dynamic Data Exchange (DDE)*, а також периферійних пристроїв, які взаємодіють безпосередньо з MATLAB. Багато з цих можливостей відомі під назвою MATLAB API.

– *COM.* Пакет MATLAB надає доступ до функцій, що дозволяють створювати, маніпулювати і видаляти COM-об'єкти (як клієнти, так і сервера). Підтримується також технологія ActiveX. Всі COM-об'єкти належать до спеціального COM-класу пакета MATLAB. Всі програми, що мають функції контролера автоматизації (англ. *Automation controller)*, можуть мати доступ до MATLAB як до сервера автоматизації (англ. *Automation server).*

– *DDE.* Пакет MATLAB містить функції, які дозволяють йому отримувати доступ до інших додатків середовища Windows, так само як і цим програмам отримувати доступ до даних MATLAB, за допомогою технології динамічного обміну даними (DDE). Кожна програма, яка може бути DDE-сервером, має своє унікальне ідентифікаційне ім'я. Для MATLAB це ім'я - *Matlab.*

5. *Веб-сервіси.* В MATLAB існує можливість викликати методи веб-сервісів. Спеціальна функція створює клас, грунтуючись на методах API веб-сервісу. MATLAB взаємодіє з клієнтом веб-сервісу за допомогою прийняття від нього повідомлень, їх обробки та відсилання відповіді. Підтримуються такі технології: Simple Object Access Protocol (SOAP) і Web Services Description Language (WSDL).

Набори інструментів

Для MATLAB є можливість створювати спеціальні набори інструментів (англ. *toolbox),* що розширюють його функціональність. Набори інструментів – це колекції функцій, написаних мовою MATLAB для вирішення певного класу задач. Компанія Mathworks постачає набори інструментів, які використовуються в багатьох галузях, охоплюючи нижченаведені.

1. Цифрова обробка сигналів, зображень та даних: *DSP Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communication Toolbox, Filter Design Toolbox* – набори функцій, які дозволяють вирішувати широкий спектр задач обробки сигналів, зображень, проектування цифрових фільтрів і систем зв'язку.

2. Системи управління: *Control Systems Toolbox, μ-Analysis and Synthesis Toolbox, Robust Control Toolbox, System Identification Toolbox, LMI Control Toolbox, Model Predictive Control Toolbox, Model-Based Calibration Toolbox* – набори функцій, що полегшують аналіз і синтез динамічних систем, проектування, моделювання та ідентифікацію систем управління,

охоплюючи сучасні алгоритми управління, такі як робастне управління, H ∞-керування, ЛМН-синтез, μ-синтез та інші.

3. Фінансовий аналіз: *GARCH Toolbox, Fixed-Income Toolbox, Financial Time Series Toolbox, Financial Derivatives Toolbox, Financial Toolbox, Datafeed Toolbox* – набори функцій, які дозволяють швидко та ефективно збирати, обробляти і передавати різну фінансову інформацію.

4. Аналіз і синтез географічних карт, в тому числі тривимірні: *Mapping Toolbox.*

5. Збір та аналіз експериментальних даних: *Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, Link for Code Composer Studio* – набори функцій, які дозволяють зберігати й обробляти дані, отримані в ході експериментів, у тому числі в реальному часі. Підтримується широкий спектр наукового та інженерного вимірювального обладнання.

6. Візуалізація та подання даних: *Virtual Reality Toolbox* – дозволяє створювати інтерактивні світи і візуалізувати наукову інформацію за допомогою технологій віртуальної реальності та мови VRML.

7. Засоби розробки: *MATLAB Builder for COM, MATLAB Builder for Excel, MATLAB Builder for NET, MATLAB Compiler, Filter Design HDL Coder* – набори функцій, які дозволяють створювати незалежні програми з середовища MATLAB.

8. Взаємодія із зовнішніми програмними продуктами: *MATLAB Report Generator, Excel Link, Database Toolbox, MATLAB Web Server, Link for ModelSim* – набори функцій, що дозволяють зберігати дані в різних форматах таким чином, щоб інші програми могли з ними працювати.

9. Бази даних: *Database Toolbox* – інструменти роботи з базами даних.

10. Наукові та математичні пакети: *Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed-Point Toolbox, Fuzzy Logic Toolbox, Genetic Algorithm and Direct Search Toolbox, OPC Toolbox, Optimization Toolbox, Partial Differential Equation Toolbox, Spline Toolbox, Statistic Toolbox, RF Toolbox* – набори спеціалізованих математичних функцій, що дозволяють вирішувати широкий спектр наукових та інженерних завдань, охоплюючи розробку генетичних алгоритмів, вирішення завдань в приватних похідних, цілочисельні проблеми, оптимізацію систем та інші.

11. Нейронні мережі: *Neural Network Toolbox* – інструменти для синтезу та аналіз нейронних мереж.

12. Нечітка логіка: *Fuzzy Logic Toolbox* – інструменти для побудови й аналізу нечітких множин.

13. Символьні обчислення: *Symbolic Math Toolbox* – інструменти для символьних обчислень з можливістю взаємодії з символьним процесором програми Maple.

Крім перерахованих вище, існують тисячі інших наборів інструментів для MATLAB, написаних іншими компаніями та ентузіастами.

Існує велика кількість програмних пакетів для вирішення задач чисельного аналізу. Багато таких пакетів є вільним програмним забезпеченням.

Сумісні з MATLAB на рівні мови програмування :

- GNU Octave GNU Octave;
- FreeMat FreeMat;
- Scilab Scilab;
- Rlab Rlab.

Близькі за функціональністю

- R , S и SPlus . R, S і SPlus;
- APL і його нащадки: наприклад J;
- Python, при використанні з такими бібліотеками, як NumPy і SciPy реалізує подібні можливості;
- IDL (англ. *Interactive Data Language,* інтерактивна мова опису даних), колись був комерційним конкурентом MATLAB, зараз залишається серйозним конкурентом в багатьох прикладних областях, хоча його частка на ринку програмних продуктів для чисельного аналізу різко впала.

Якщо є необхідність розробки великих проектів для чисельного аналізу, то можливе використання мов програмування загального призначення, що підтримують статичну типізацію та модульну структуру. Прикладами можуть служити Modula-3, Haskell, Ада, Java. При цьому рекомендується використовувати відомі у науково-інженерному середовищі спеціалізовані бібліотеки.

В даний час пакет MATLAB являє собою розвинене інтегральне програмне середовище, що має власну мову програмування. Він дає користувачеві можливість швидко виконуєти різні операції над векторами і матрицями, такі як множення і обернення матриць, обчислення визначників, находження власних чисел і векторів. Крім того, в MATLAB входять операції обчислення звичайних функцій (алгебраїчних, тригонометричних, логічних), розв'язання алгебраїчних і дмференціальних рівнянь, операції побудови графіків і ряд інших.

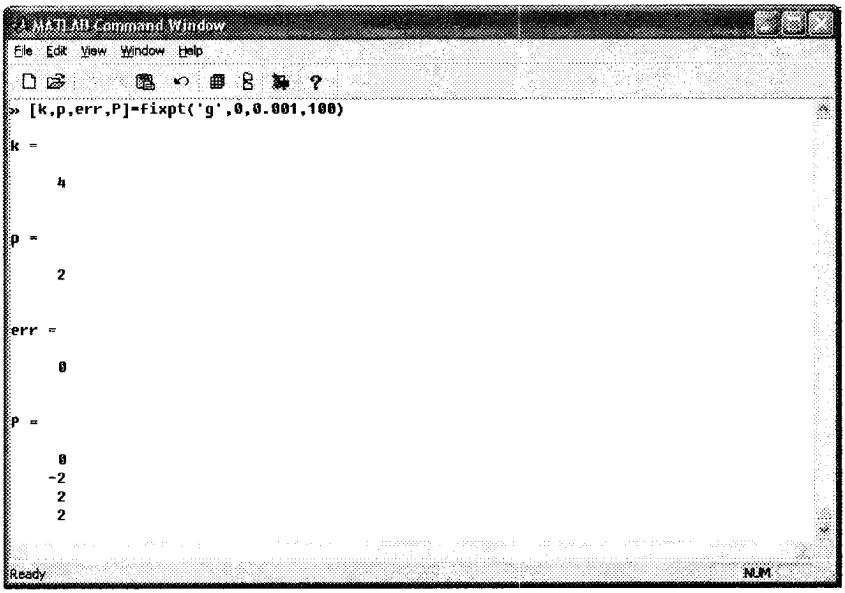
Нелінійні рівняння:

```
function [k,p,err,P]=fixpt(g,p0,tol,max1)
% Розвязання рівнянь ітераційним методом
% Введення - g - ітераційна фунція, що вводиться як рядок 'g'
%                p0 - початкове наближення
%                tol - допустиме відхилення
%                max1 - максимальна кількість ітерацій
% Вихід   - k - число виконаних ітерацій
%                p - наближення для точки
%                err - похибка наближення
%                P - містить послідовність {pn}
P(1)=p0;
for k=2:max1
  P(k)=feval(g,P(k-1));
  err=abs(P(k)-P(k-1));
  relerr=err/(abs(P(k))+eps);
  p=P(k);
  if (err<tol) | (relerr<tol),break,end
end
if k==max1
  disp('максимально допустима кількість ітерацій')
end
P=P';
```

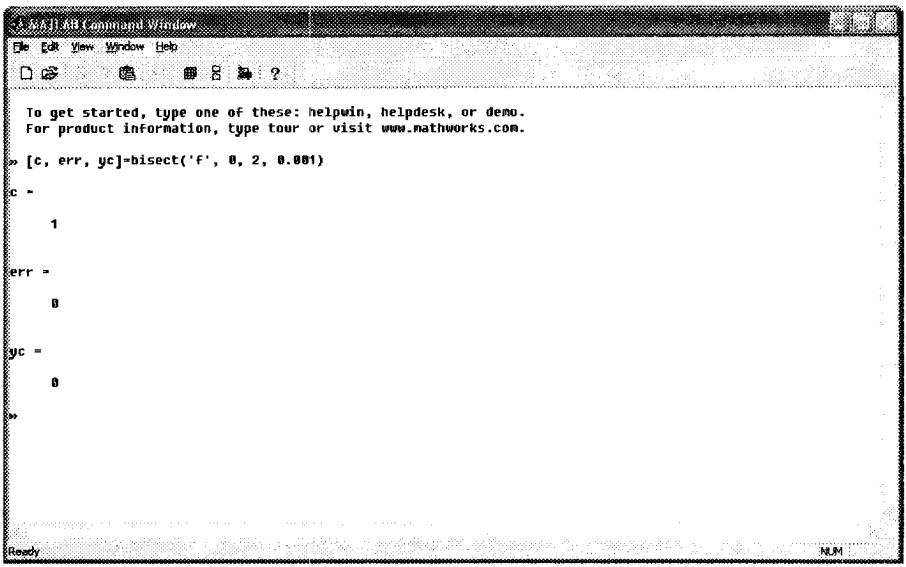
Щоб запустити програму необхідно у MATLAB Command Window записати

[c, err, yc]=bisect('f', a, b, delta),

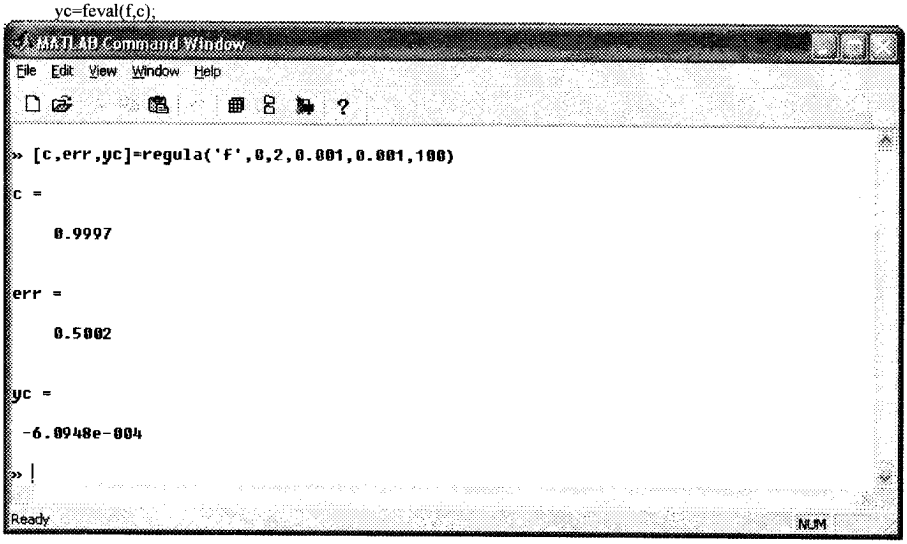
де замість a, b, delta поставити необхідні значення. При цьому файл програми bisect.m та функції f.m повинні знаходитись у папці work папки, де встановлений Matlab.



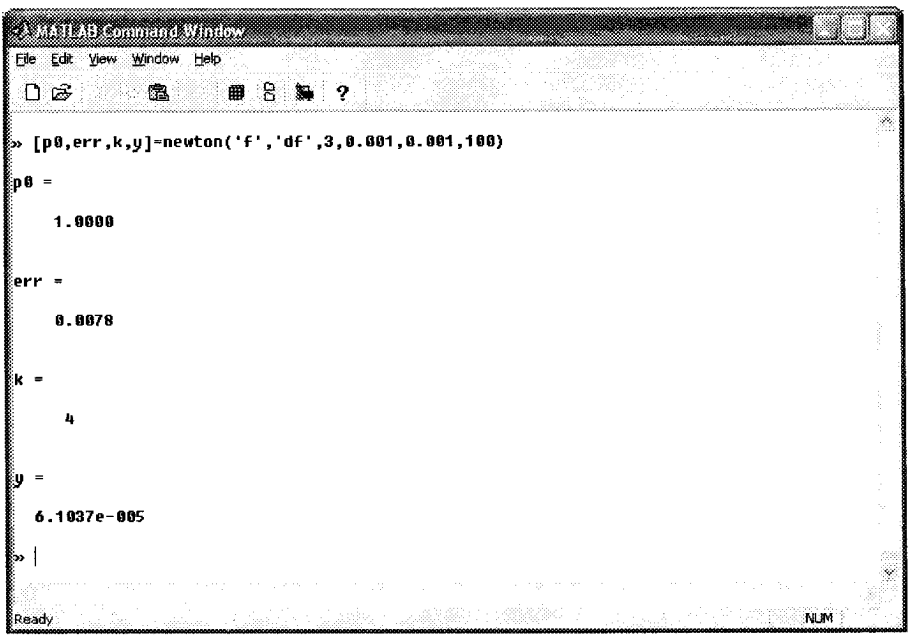

```
function [c, err, yc]=bisect(f, a, b, delta)
% Метод половинного ділення для розв'язання рівнянь
% Введення - f - вводиться як рядок 'f'
% a і b - ліва та права крайні точки
% delta - допустиме відхилення
% Вихід - c - нуль
% yc=f(c)
% err - помилка обчислення c
ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0,break,end
max1=1+round((log(b-a)-log(delta))/log(2));
for k=1:max1
  c=(a+b)/2;
  yc=feval(f,c);
  if yc==0
    a=c;
    b=c;
  elseif yb*yc>0
      b=c;
      yb=yc;
    else
      a=c;
      ya=yc;
    end
    if b-a < delta, break, end
  end
  c=(a+b)/2;
  err=abs(b-a);
  yc=feval(f,c);
```


MATLAB Command Window
File Edit View Window Help
To get started, type one of these: helpwin, helpdesk, or demo.
For product information, type tour or visit www.mathworks.com.
» [c, err, yc]=bisect('f', 0, 2, 0.001)
c =
1
err =
0
yc =
0
»
Ready
NUM

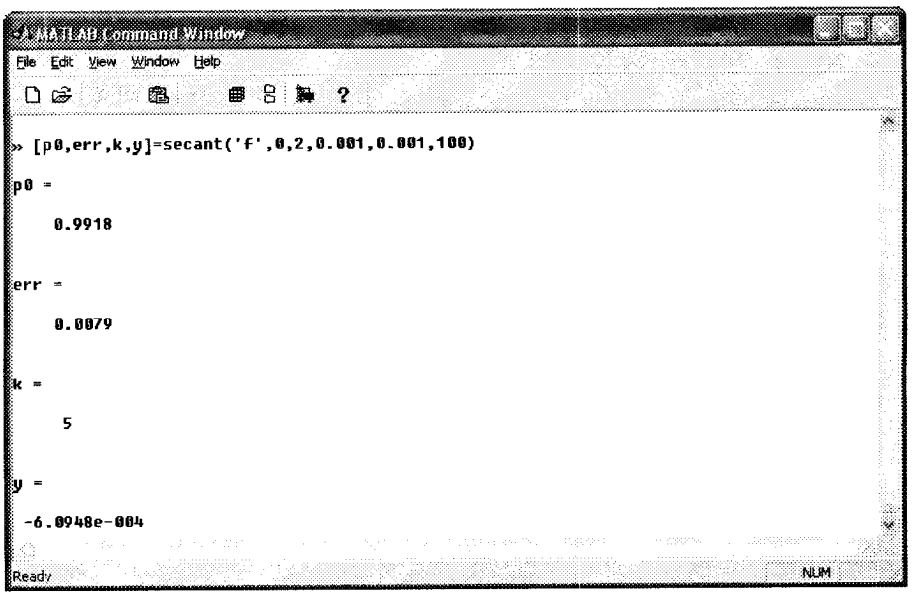
```
function [c,err,yc]=regula(f,a,b,delta,epsilon,max1)
% Метод хорд (хибного положення) для розв'язання рівнянь
% Введення - f - функція, що вводиться як рядок 'f'
%            a и b - ліва та права крайні точки
%            delta - допустиме відхилення р0
%               epsilon - допустиме відхилення для значення функції у
%               max1 - максимальна кількість ітерацій
% Вихід     - c - нуль
%               yc=f(c)
%               err - похибка обчислення для с
ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0
  disp('Зауваження: f(a)*f(b)>0'),
  break,
end
for k=1:max1
  dx=yb*(b-a)/(yb-ya);
  c=b-dx;
  ac=c-a;
  yc=feval(f,c);
  if yc==0,break;
  elseif yb*yc>0
    b=c;
    yb=yc;
  else
    a=c;
    ya=yc;
  end
  dx=min(abs(dx),ac);
  if abs(dx)<delta,break,end
  if abs(yc)<epsilon,break,end
end
err=abs(b-a)/2;
yc=feval(f,c);
```



```
function [p0,err,k,y]=newton(f,df,p0,delta,epsilon,max1)
% Метод Ньютона для розв'язання рівнянь
% Введення - f - функція, що вводиться як рядок 'f'
%            df - похідна f, що вводиться як рядок 'df'
%            p0 - початкове наближення f до нуля
%            delta - допустиме відхилення p0
%            epsilon - допустиме відхилення для значення функції y
%            max1 - максимальна кількість ітерацій
% Вихід      p0 - наближення Ньютона до нуля
%            err - похибка обчислень для p0
%            k - кількість ітерацій
%            y - значення функції f(p0)
for k=1:max1
  p1=p0-feval(f,p0)/feval(df,p0);
  err=abs(p1-p0);
  relerr=2*err/(abs(p1)+delta);
  p0=p1;
  y=feval(f,p0);
  if (err<delta)|(relerr<delta)|(abs(y)<epsilon),break,end
end
```

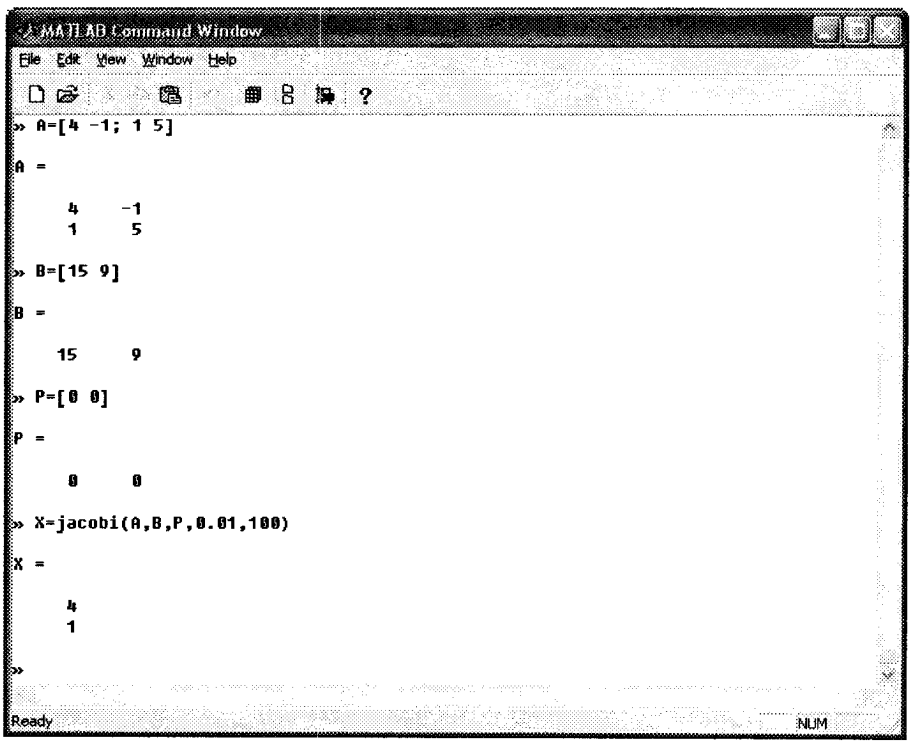


```
function [p0,err,k,y]=secant(f,p0,p1,delta,epsilon,max1)
% Метод січних для розв'язання рівнянь
% Введення - f - функція, що вводиться як рядок 'f'
%                p0 і p1 - початкове наближення f до нуля
%                   delta - допустиме відхилення p1
%                   epsilon - допустиме відхилення для значення функції y
%                   max1 - максимальна кількість ітерацій
% Вихід          p1 - наближення до нуля методу січних
%                   err - похибка обчислень для p1
%                   k - кількість ітерацій
%                   y - значення функції f(p1)
for k=1:max1
  p2=p1-feval(f,p1)*(p1-p0)/(feval(f,p1)-feval(f,p0));
  err=abs(p1-p2);
  relerr=2*err/(abs(p2)+delta);
  p0=p1;
  p1=p2;
  y=feval(f,p1);
  if (err<delta)|(relerr<delta)|(abs(y)<epsilon),break,end
end
```

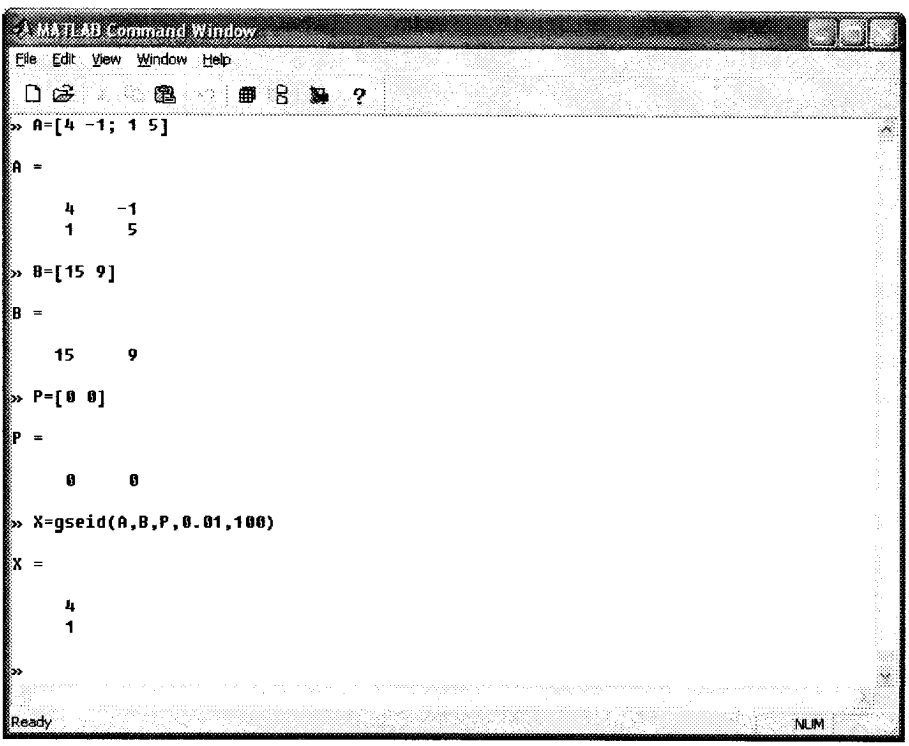


Системи рівнянь:

```
function X=jacobi(A,B,P,delta,max1)
% Розв'язання лінійних рівнянь методом ітерації Якобі
% Введення - A - невироджена матриця розміром N×N
%            B - матриця розміром N×1
%            P - матриця розміром N×1, початкові наближення
%            delta - допустиме відхилення для P
%            max1 - максимальне число відхилень
% Вихід     - X - матриця розміром N×1, наближення Якобі до розв'язку
%            AX=B
N=length(B);
for k=1:max1
  for j=1:N
    X(j)=(B(j)-A(j,[1:j-1,j+1:N])*P([1:j-1,j+1:N]))/A(j,j);
  end
  P=X';
end
X=X';
```

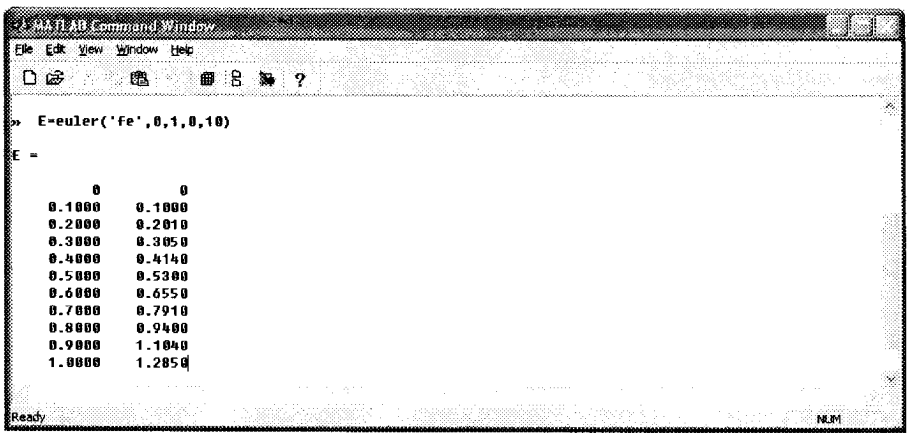


```
function X=gseid(A,B,P,delta,max1)
    % Розв'язання лінійних рівнянь методом ітерації Якобі
    % Введення - A - невироджена матриця розміром N×N
    %            B - матриця розміром N×1
    %            P - матриця розміром N×1, початкові наближення
    %            delta - допустиме відхилення для P
    %            max1 - максимальне число відхилень
    % Вихід     - X - матриця розміром N×1, наближення Якобі до розв'язку
    %            AX=B
    N=length(B);
    for k=1:max1
      for j=1:N
        if j==1
          X(1)=(B(1)-A(1,2:N)*P(2:N))/A(1,1);
        elseif j==N
            X(N)=(B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
            else
            % X містить k-е наближення і P(k-1)
            X(j)=(B(j)-A(j,1:j-1)*X(1:j-1)-A(j,j+1:N)*P(j+1:N))/A(j,j);
        end
      end
      P=X';
      end
    X=X';
```

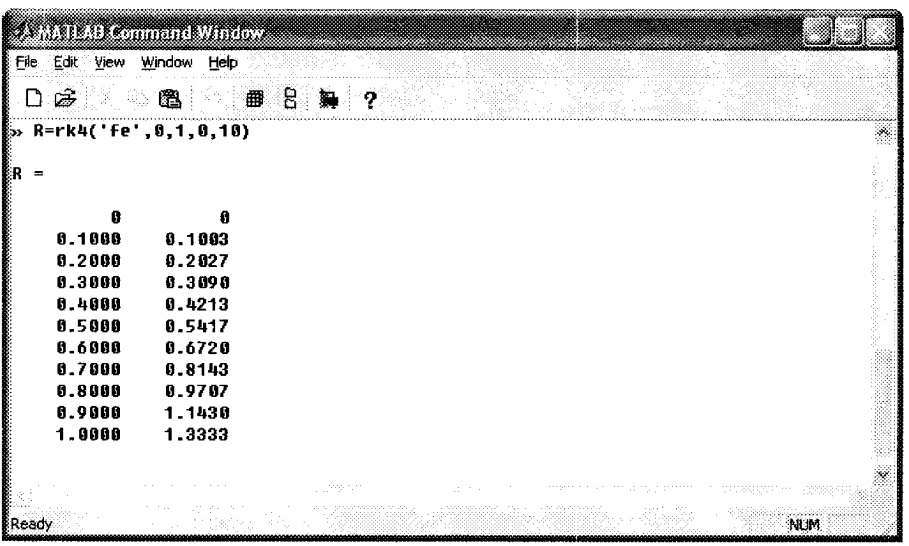


Диференціальні рівняння:

```
function E=euler(f,a,b,ya,M)
% Розв'язання диференціального рівняння прямим методом Ейлера
% Вхід  - f - функція, що вводиться, як рядок 'f'
%       - a і b - ліва і правая крайні точки
%       - ya - початкова умова y(a)
%       -  M - кількість кроків
% Вихід - E=[T' Y'], де T - вектор абсцис
%       - Y - вектор ординат
h=(b-a)/M;
T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
  Y(j+1)=Y(j)+h*feval(f,T(j),Y(j));
end
E=[T' Y'];
```

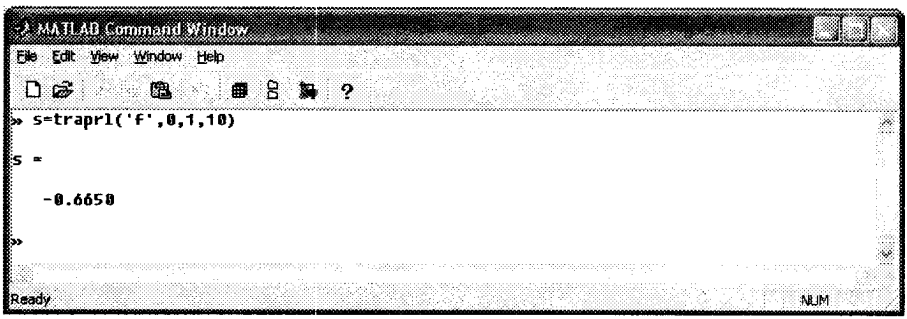


```
function R=rk4(f,a,b,ya,M)
% Розв'язання диференціальних рівнянь методом Рунге-Кутта
% Вхід  - f - функція, що вводиться, як рядок 'f'
%       - a і b - ліва і правая крайні точки
%       - ya - початкова умова y(a)
%       - M - кількість кроків
% Вихід - R=[T' Y'], де T - вектор абсцис
%       - Y - вектор ординат
h=(b-a)/M;
T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
  k1=h*feval(f,T(j),Y(j));
  k2=h*feval(f,T(j)+h/2,Y(j)+k1/2);
    k3=h*feval(f,T(j)+h/2,Y(j)+k2/2);
    k4=h*feval(f,T(j)+h,Y(j)+k3);
    Y(j+1)=Y(j)+(k1+2*k2+2*k3+k4)/6;
end
R=[T' Y'];
```

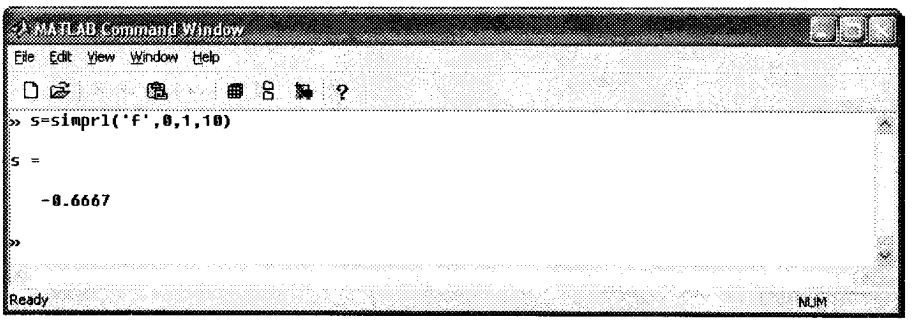


Інтеграли:

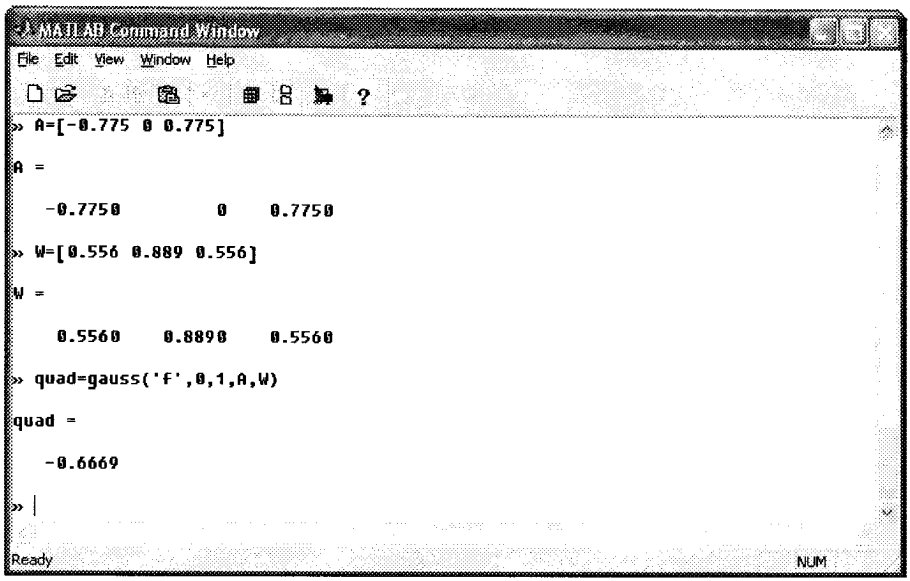
```
function s=traprl(f,a,b,M)
% Знаходження значення інтеграла методом трапецій
% Введення - f - підінтегральна функція, що вводиться як рядок 'f'
%          - a і b - верхня та нижня границі інтегрування
%          - M - кількість підінтервалів
% Вихід    - s - сума формули трапецій
h=(b-a)/M;
s=0;
for k=1:(M-1)
  x=a+h*k;
  s=s+feval(f,x);
end
s=h*(feval(f,a)+feval(f,b))/2+h*s;
```



```
function s=simprl(f,a,b,M)
% Знаходження значення інтеграла методом Сімпсона
% Введення - f - підінтегральна функція, що вводиться як рядок 'f
%          - a і b - верхня та нижня границі інтегрування
%          - M - кількість підінтервалів
% Вихід    - s - сума формули Сімпсона
h=(b-a)/(2*M);
s1=0;
s2=0;
for k=1:M
  x=a+h*(2*k-1);
  s1=s1+feval(f,x);
end
for k=1:(M-1)
  x=a+h*2*k;
  s2=s2+feval(f,x);
end
s=h*(feval(f,a)+feval(f,b)+4*s1+2*s2)/3;
```

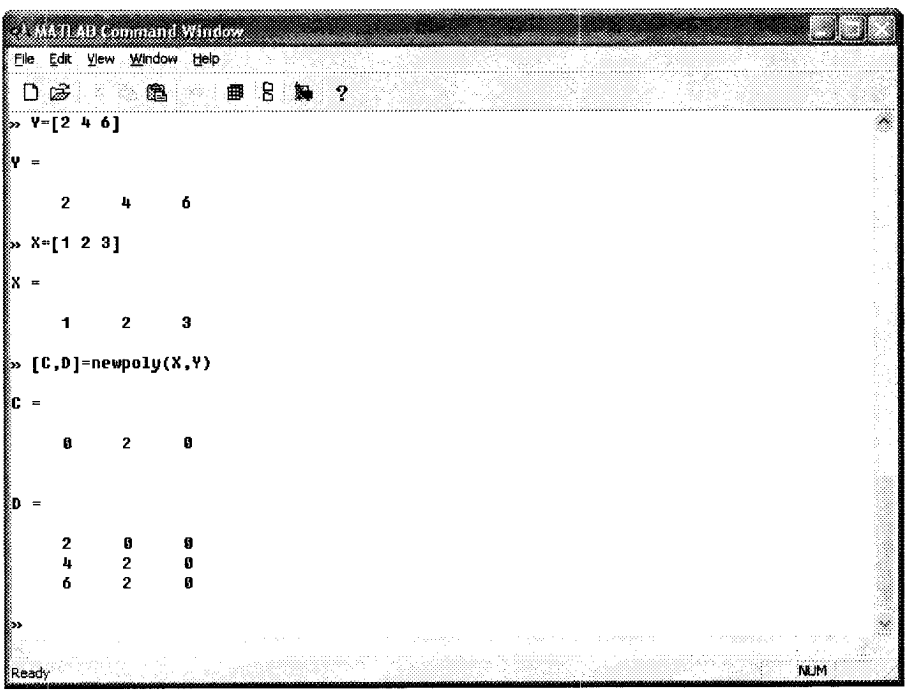


```
function quad=gauss(f,a,b,A,W)
% Знаходження значення інтеграла методом Гаусса
% Введення - f - підінтегральна функція, що вводиться як рядок 'f'
%          - a і b - верхня та нижня границі інтегрування
%          - A - вектор абсцис розміром 1×N (вводиться з таблиці абсцис і ваг для формули Гаусса)
%          - W - вектор ваг розміром 1×N (вводиться з таблиці абсцис і ваг для формули Гаусса)
% Вихід    - quad - значення інтеграла
N=length(A);
T=zeros(1,N);
T=((a+b)/2)+((b-a)/2)*A;
quad=((b-a)/2)*sum(W.*feval(f,T));
```

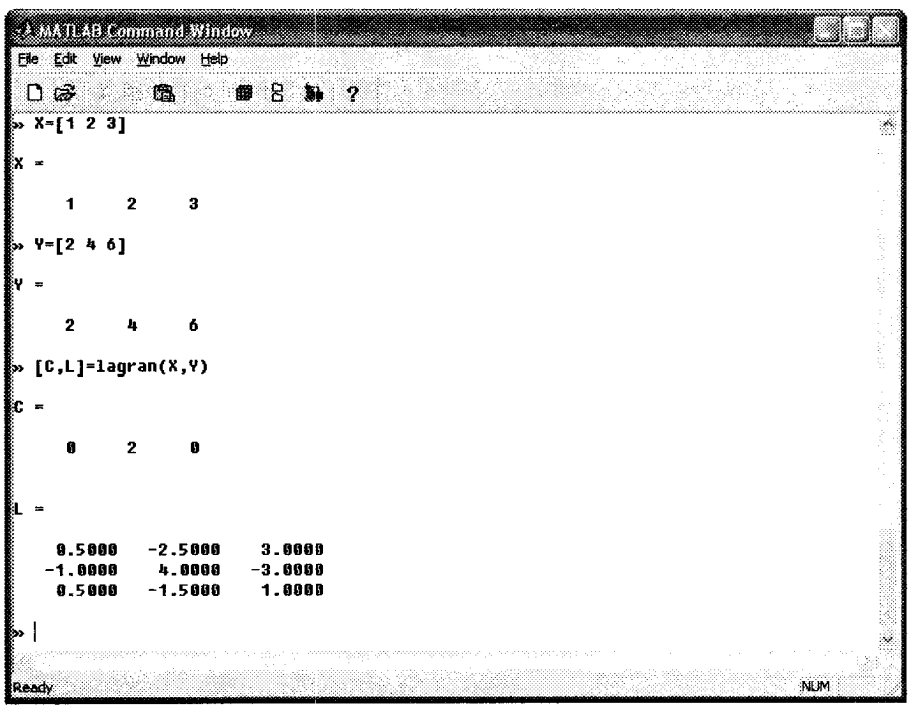


Інтерполяція:

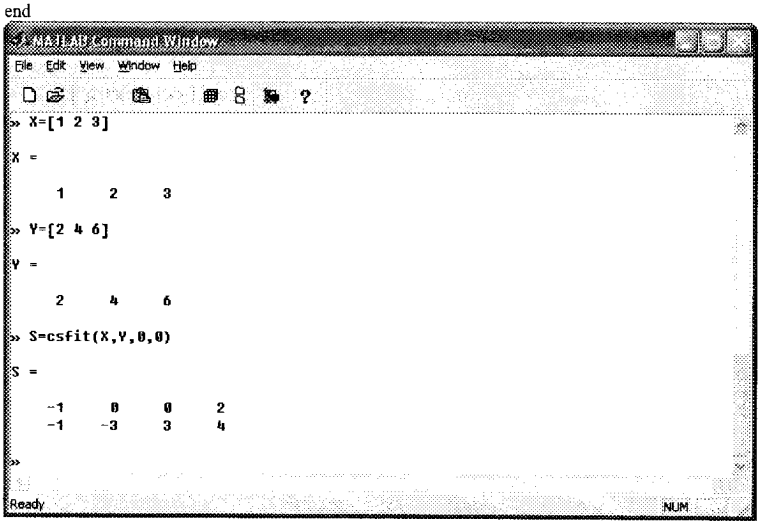
```
function [C,D]=newpoly(X,Y)
% Інтерполяція методом Ньютона
% Введення - X - вектор абсцис
%          - Y - вектор ординат
% Вихід    - C - вектор коефіцієнтів полінома Ньютона
%          - D - таблиця різницевих відношень
n=length(X);
D=zeros(n,n);
D(:,1)=Y';

% Формування таблиці різниць
for j=2:n
  for k=j:n
    D(k,j)=(D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
  end
end

% Визначення коефіцієнтів полінома Ньютона
C=D(n,n);
for k=(n-1):-1:1
  C=conv(C,poly(X(k)));
  m=length(C);
  C(m)=C(m)+D(k,k);
end
```



```
function [C,L]=lagran(X,Y)
% Інтерполяція методом Лагранжа
% Введення - X - вектор абсцис
%          - Y - вектор ординат
% Вихід    - C - вектор коефіцієнтів полінома Лагранжа
%          - L - матриця коефієнтів полінома Лагранжа
w=length(X);
n=w-1;
L=zeros(w,w);

% Формування коефіцієнтів полінома Лагранжа
for k=1:n+1
  V=1;
  for j=1:n+1
    if k~=j
    V=conv(V,poly(X(j)))/(X(k)-X(j));
  end
end
   L(k,:)=V;
end

% Визначення коефіцієнтів полінома Лагранжа
C=Y*L;
```

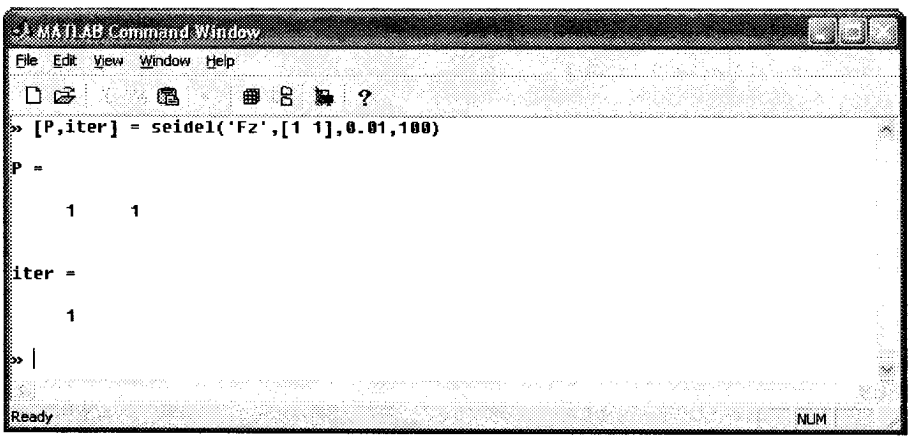


```
function S=csfit(X,Y,dx0,dxn)
% Інтерполяція кубічними сплайнами
%Введення - X - вектор абсцис розміром 1×n
%          - Y - вектор ординат розміром 1×n
%          - dx0 = S'(x0) - гранична умова на першу похідну
%          - dxn = S'(xn) - гранична умова на першу похідну
% Вихід    - S: рядки S - це коефіцієнти в порядку спадання
%            для кубічного сплайна
N=length(X)-1;
H=diff(X);
D=diff(Y)./H;
A=H(2:N-1);
B=2*(H(1:N-1)+H(2:N));
C=H(2:N);
U=6*diff(D);

B(1)=B(1)-H(1)/2;
U(1)=U(1)-3*(D(1)-dx0);
B(N-1)=B(N-1)-H(N)/2;
U(N-1)=U(N-1)-3*(dxn-D(N));
for k=2:N-1
  temp=A(k-1)/B(k-1);
  B(k)=B(k)-temp*C(k-1);
  U(k)=U(k)-temp*U(k-1);
end
M(N)=U(N-1)/B(N-1);
for k=N-2:-1:1
  M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
end
M(1)=3*(D(1)-dx0)/H(1)-M(2)/2;
M(N+1)=3*(dxn-D(N))/H(N)-M(N)/2;
for k=0:N-1
  S(k+1,1)=(M(k+2)-M(k+1))/(6*H(k+1));
  S(k+1,2)=M(k+2)/2;
  S(k+1,3)=D(k+1)-H(k+1)*(2*M(k+1)+M(k+2))/6;
  S(k+1,4)=Y(k+1);
end
```



Системи нелінійних рівнянь:

```
function [P,iter] = seidel(G,P,delta,max1)
% Розв'язання систем нелінійних рівнянь ітерацією Зейделя
% Введення - G - нелійна система, записана в формі M-файла G.m
%          - P - початкове наближення до розв'язку
%          - delta - можлива похибка
%          - max1 - число ітерацій
% Вихід    - P - наближення Зейделя до розв'язку
%          - iter - число виконаних ітерацій
N=length(P);
for k=1:max1
  X=P;
  for j=1:N
    A=feval('G',X);
    X(j)=A(j);
  end
  err=abs(norm(X-P));
  relerr=err/(norm(X)+eps);
  P=X;
  iter=k;
  if(err<delta)|(relerr<delta)
    break
  end
end
```



Функції, що використовувались для запуску даних програм:

```
function y=f(x)
y=x.^2-1;

function y=df(x)
y=2*x;

function y=fe(x,z)
y=x.^2+1;

function y=g(x)
```

```
y=x.^2-2;

function Z=Fz(X)
x=X(1);
y=X(2);
Z=zeros(1,2);
Z(1)=x.^4;
Z(2)=y.^2;
```

Можливості цифрової обробки зображень в Matlab

На сьогоднішній день система Matlab, зокрема пакет прикладних програм Image Processing Toolbox, є найбільш потужним інструментом для моделювання й дослідження методів обробки зображень. Він містить велику кількість вбудованих функцій, що реалізують найпоширеніші методи обробки зображень. Розглянемо основні можливості пакета Image Processing Toolbox.

Пакет Image Processing надає широкий спектр засобів для цифрової обробки та аналізу зображень. Бувши тісно пов'язаним із середовищем розробки додатків MATLAB, пакет Image Processing Toolbox звільняє користувача від виконання тривалих операцій кодування та налагодження алгоритмів, дозволяючи зосередити зусилля на вирішенні основного наукового чи практичного завдання.

Основні властивості пакета:

- відновлення й виділення деталей зображень;
- робота з виділеною ділянкою зображення;
- аналіз зображення;
- лінійна фільтрація;
- перетворення зображень;
- геометричні перетворення;
- збільшення контрастності важливих деталей;
- бінарні перетворення;
- обробка зображень і статистика;
- колірні перетворення;
- зміна палітри;
- перетворення типів зображень.

Пакет Image Processing дає широкі можливості для створення й аналізу графічних зображень у середовищі MATLAB. Цей пакет забезпечує надзвичайно гнучкий інтерфейс, що дозволяє маніпулювати зображеннями, інтерактивно розробляти графічні картини, візуалізувати набори даних і анотувати результати для технічних описів, доповідей і публікацій.

Пакет Image Processing інтенсивно використовується в більш ніж 4000 компаніях і університетах по всьому світі. При цьому є дуже широке коло завдань, які користувачі вирішують за допомогою даного пакета,

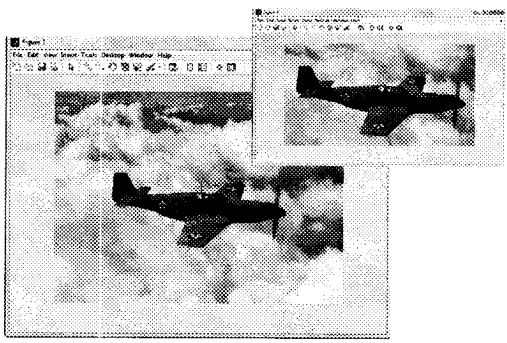
наприклад космічні дослідження, військові розробки, астрономія, медицина, біологія, робототехніка, матеріалознавство, генетика й т. д. Він містить велику кількість вбудованих функцій, що реалізують найпоширеніші методи обробки зображень. Розглянемо основні можливості пакета Image Processing Toolbox.

Геометричні перетворення зображень

До найпоширеніших функцій геометричних перетворень належить кадрування зображень (imcrop), зміна розмірів (imresize) і поворот зображення (imrotate).

Суть кадрування полягає в тому, що функція imcrop дозволяє за допомогою миші в інтерактивному режимі вирізати частину зображення та помістити його в новому вікні перегляду.

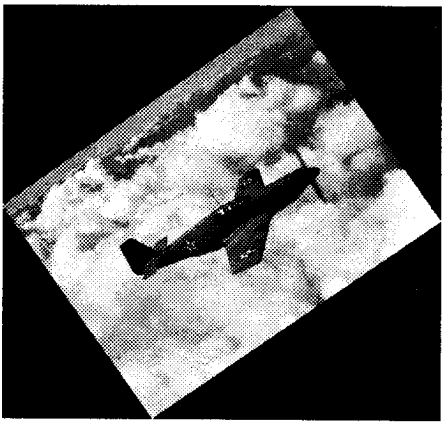
```
L=imread('airplane.tif');
imshow(L);
imcrop;
```



Застосування функцій imshow та imcrop

У пакеті Image Processing Toolbox існує функція imrotate, що здійснює поворот зображення на заданий кут.

```
L1=imrotate(L,35,'bicubic');
figure,imshow(L1)
```



Застосування функції imrotate

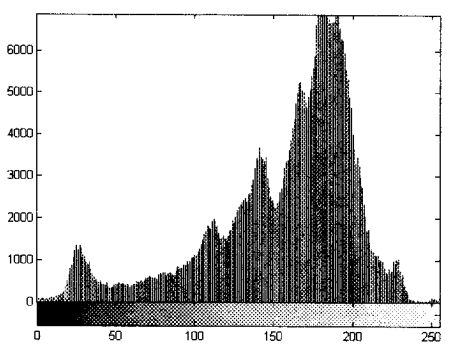
Таким чином, наведені вище функції дозволяють повертати, вирізати частини, масштабувати, тобто працювати з цілим масивом зображення.

Аналіз зображень

Для роботи з окремими елементами зображень використовуються такі функції, як imhist, impixel, mean2, corr2 та інші.

Однією з найбільш важливих характеристик зображення є гістограма розподілу значень інтенсивностей пікселів зображення, яку можна побудувати за допомогою функції imhist.

```
L=imread('cameraman.tif');
figure, imshow(L);
figure, imhist(L);
```



Застосування функції imhist

Досить часто при проведенні аналізу зображень виникає необхідність визначити значення інтенсивностей деяких пікселів. Для цього в інтерактивному режимі можна використовувати функцію impixel.

```
L=imread('Air015.jpg');
c = [12 146 410];
r = [104 156 129];
pixels=impixel(L,c,r)

pixels =

  190   190   190
  187   187   187
  202   202   202
```

Ще однією поширеною функцією є функція mean2 – вона обчислює середнє значення елементів матриці.

```
B=mean2(L)

B =

          156.6090
```

Функція corr2 обчислює коефіцієнт кореляції між двома матрицями. Інакше кажучи, за допомогою функції corr2 можна сказати наскільки дві матриці зображення схожі між собою. Ця функція широко застосовується при вирішенні завдань розпізнавання.

Усунення розмитості зображення за допомогою фільтра Віннера

Віннерівська деконволюція може бути ефективно використана, якщо частота зображення та адитивний шум відомі. Як приклад використаємо рисунок В.1, а).

```
A=imread('C:\\OLYA\NEW_book\Hibiscus2.jpg');
imshow(A)
```

Зімітуємо розмиття зображення, яке може виникнути під час руху камери. Створимо функцію розсіювання точки (point-spread function – PSF), яка відповідає лінійному зсуву вздовж 31 пікселя (LEN=31), при куті нахилу в 11 градусів (THETA=11). Розмите зображення наведено на рисунку В.1, б)

```
LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA);
blurred = imfilter(I,PSF,'circular','conv');
```

```
figure, imshow(blurred);
```

Для того, щоб підтвердити важливість визначення істинного значення PSF, виконаємо три етапи відновлення. Для першого відновлення wnr1 використаємо PSF, створену на попередньому кроці (рисунок В.1, в)

```
wnr1 = deconvwnr(blurred,PSF);
figure, imshow(wnr1);
```

Для кращої наочності добавимо адитивний шум (рисунок В.1, г)

```
noise = 0.1*randn(size(I));
blurredNoisy = imadd(blurred,im2uint8(noise));
figure, imshow(blurredNoisy);
```

Відновимо розмите та зашумлене зображення, використавши інверсну фільтрацію та порівняємо з рисунком В. 1,в. Але врахуємо, що в цьому випадку зображення було не лише розмите, але й зашумлене (рисунок В.1, д)

```
wnr4 = deconvwnr(blurredNoisy,PSF);
figure, imshow(wnr4);
```

Для контролю посилення шуму передбачимо значення відношення шум – сигнал (noise-to-signal power ratio – NSR), рисунок В.1, е

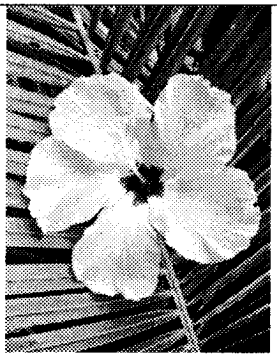
```
NSR = sum(noise(:).^2)/sum(im2double(I(:)).^2);
wnr5 = deconvwnr(blurredNoisy,PSF,NSR);
figure, imshow(wnr5);
```

Змінимо значення NSR для покращення результатів відновлення. Мале значення NSR value посилює шум (рисунок В.1, ж)

```
wnr6 = deconvwnr(blurredNoisy,PSF,NSR/2);
figure, imshow(wnr6);
```

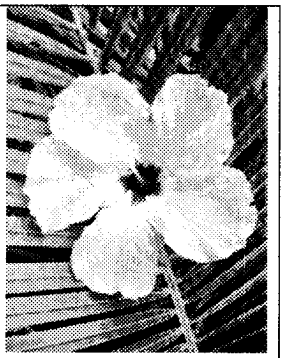
На останньому кроці покращення результатів відновлення зображення застосуємо автокореляційну функцію до шуму NCOR та до сигналу ICOR (рисунок В.1, и)

```
NP = abs(fftn(noise)).^2;
NPOW = sum(NP(:))/prod(size(noise)); % сила шуму
NCOR = fftshift(real(ifftn(NP)));
IP = abs(fftn(im2double(A))).^2;
IPOW = sum(IP(:))/prod(size(A));
ICOR = fftshift(real(ifftn(IP)));
wnr7 = deconvwnr(blurredNoisy,PSF,NCOR,ICOR);
figure, imshow(wnr7);
```

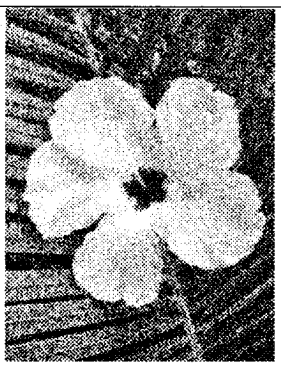


а) Вихідне зображення

б) Розмите зображення

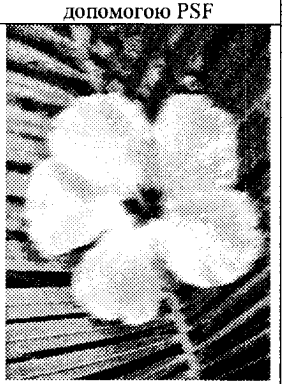


в) Відновлене за допомогою PSF

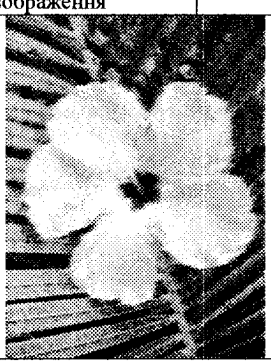


г) Розмите та зашумлене зображення

д) Інверсна фільтрація



е) Відновлене з NSR



ж) Відновлене з NSR/2

и) Застосування автокореляційної функції

Рисунок В.1 – Усунення розмитості зображення

ДОДАТОК Г

CONCISE ENGLISH-RUSSIAN-UKRAINIAN-POLISH DICTIONARY OF TERMS

A

e **absolute congervence** r абсолютная сходимость; u абсолютна збіжність; p absolutna zbieżność

e **absolute error** r абсолютная погрешность; u абсолютна похибка; p błąd bezwzględny

e **acceptable variation** r допустимое отклонение; u припустиме відхилення; p dopuszczalne odchylenie

e **adaptive algorithm** r адаптивный алгоритм; u адаптивний алгоритм; p adaptacyjny algorytm

e **admissible error** r допустимая ошибка; u припустима помилка; p dopuszczalny błąd

e **accidental variable** r случайная переменная; u випадкова змінна; p okazyjna zmienna

e **accuracy of estimate** r точность оценки; u точність оцінки; p precyzja oceny

e **algebraic equation** r алгебраическое уравнение; u алгебраїчне рівняння; p równanie algebraiczne

e **algorithm convergence** r сходимость алгоритма; u збіжність алгоритму; p zbieżność algorytmu

e **analog-to-digital conversion** r аналого-цифровое преобразование; u аналого-цифрове перетворення; p konwersja analogowo cyfrowa

e **antiderivative function** r первообразная функция; u первісна функція; p funkcja pierwotna

e **approach** r приближение; u наближення; p zbliżać sie

e **approximation** r аппроксимация; u апроксимація; p aproksymacja

e **appropriate behavior function** r регулярная функция; u регулярна функція; p regularna funkcja

e **artificial intelligence** r искусственный интеллект; u штучний інтелект; p sztuczny intelekt

e **autocorrelation function** r функция автокорреляции; u функція автокореляції; p funkcja autokorelacja

e **autoregressive moving average** r авторегрессионное скользящее среднее; u авторегресійне ковзне середнє; p autoregresja ślizgające się średnie

e **axis** r ось; u вісь; p oś

e **axis of the ordinates** r ось ординат; u вісь ординат; p oś rzędnych

B

e **background image** r фоновое изображение; u фонове зображення; p tło obraz

e **bandwidth** r диапазон частот; u діапазон частот; p zakres częstotliwości

e **backward difference** r левая разность; u ліва різниця; p różnica wsteczna funkcji

e **benchmark** r контрольная точка; u контрольна точка ; p kontrolny punkt

e **Bézier curve** кривая Безье r; u крива Без'є; p krzywa Beziera

e **boundary-value problem** r краевая задача; u крайова задача; p zagadnienie brzegowe

e **branched r** разветвлённый; u розгалужений; p rozwidlony

e **brightness signal** r сигнал яркости; u сигнал яскравості; p luminancja sygnału

e **bulge of the function** r выпуклость функции; u випуклість функції; p wypukłość funkcji

C

e **Cauchy problem** r задача Коши; u задача Коші; p zagadnienie Cauchy'ego

e **central difference** r центральная разность; u центральна різниця; p różnica centralna

e **column** r столбец; u стовпець; p kolumna

e **compression** r сжатие; u стиск; p kompresja
e **complex conjugate** r комплексно сопряжённые собственные значения; u комплексно-спряжені власні значення; p kompleksowy związane własne znaczenia
e **complex roots** r комплексные корни; u комплексні корені; p skomplikowane korzenie
e **complex variable function** r комплексная функция; u комплексна функція; p kompleksowa funkcja
e **computational experiment** r вычислительный эксперимент; u обчислювальний експеримент; p eksperyment obliczeniowy
e **computational scheme** r вычислительная схема; u обчислювальна схема; p obchislyuvalna schemat
e **continuous model** r непрерывная модель; u неперервна модель; p model ciągły
e **convergence** r сходимость; u збіжність; p zbieżność
e **convolution** r свертка; u згортка; p kompresuj
e **correlation** r корреляция; u кореляція; p korelacja
e **curve** r кривая; u крива; p krzywa
e **covariance** r ковариация; u коваріація; p kowariancja

D

e **data processing** r обработка данных; u обробка даних; p przetwarzanie danych
e **decomposition** r разложение; u розкладання; p rozkład
e **derivative** r производная; u похідна; p pochodny
e **determination** r детерминированность (определенность); u детермінованість (визначеність); p wyznaczanie, określanie
e **deterministic models** r детерминированные модели; u детерміновані моделі; p deterministyczne modele
e **digital filter** r цифровой фильтр; u цифровий фільтр; p cyfrowy filtr
e **digital signal processing** r цифровая обработка сигнала; u цифрова обробка сигналу; p cyfrowego przetwarzania sygnałów
e **digital simulation** r цифровое моделирование; u цифрове моделювання; p symulacja cyfrowa
e **difference** r разность; u різниця; p różnica
e **difference derivative** r разностная производная; u різницева похідна; p pochodna różnicy
e **differential equations** r дифференциальные уравнения; u диференціальні рівняння; p równania różniczkowe
e **discrete signal** r дискретный сигнал; u дискретний сигнал; p sygnał dyskretny
e **discrete exponential function** r дискретная экспоненциальная функция; u дискретна експоненціальна функція; p dyskretna funkcja wykładnicza
e **dispersion** r рассеивание; u розсіювання; p dyspersja
e **distribution error** r ошибка распространения; u похибка розповсюдження; p pomyłka dystrybucji
e **digitization** r дискретизация; u дискретизація; p dyskretyzacja
e **distribution law** r закон распределения; u закон розподілення; p prawo podziału
e **divergence** r расходимость; u розбіжність; p rozbieżność, dywergencja
e **domain boundary** r граница области; u границя області; p granica obwodu
e **dynamic range** r динамический диапазон; u динамічний діапазон; p zakres dynamiki
e **dynamic models** r динамические модели; u динамічні моделі; p dynamiczne modele

E

e **eigenvalue** r собственное значение; u власне значення; p własne znaczenie
e **eigenvector** r собственный вектор; u власний вектор; p własny wektor
e **equation** r уравнение; u рівняння; p równanie
e **elliptic equation** r эллиптическое уравнение; u еліптичне рівняння; p eliptyczne równanie
e **error** r ошибка; u похибка; p pomyłka

e **estimation** r оценивание; u оцінювання; p ocenianie
e **exactness** r точность; u точність; p ścisłość, dokładność
e **exception** r исключение; u виняток; p wyjątek
e **expectation** r ожидание; u очікування; p wartość oczekiwana

F

e **factorize** r разлагать на множители; u розкладати на множники; p deprawować na mnożniki
e **fast Fourier transform** r быстрое преобразование Фурье; u швидке перетворення Фур'є; p szybka transformacja Fouriera
e **faithful number** r правильное число; u правильне число; p numer wierny
e **finite difference** r конечная разность; u кінцева різниця; p różnica ograniczona
e **finite impulse response filter** r фильтр с конечной импульсной характеристикой; u фільтр з кінцево-імпульсною характеристикою; p filtr z końcową impulsową charakterystyką
e **firmness (stability)** r устойчивость; u стійкість; p stabilność, stałość, trwałość
e **forward difference** r правая разность; u права різниця; p różnica progresywna funkcji
e **flowgraph** r граф-схема (алгоритма); u граф-схема; p graph-program
e **frequeny response** r частотная характеристика; u частотна характеристика; p pasmo przenoszenia
e **function of belonging** r функция принадлежности; u функція належності; p przynależna funkcja
e **fuzzy logic** r нечеткая логика; u нечітка логіка; p logika rozmyta

G

e **Gaussian random process** r Гауссовский случайный процесс; u Гауссовий випадковий процес; p Gaussian proces losowy
e **geometrical constraint** r геометрическое ограничение; u геометричне обмеження; p geometryczne ograniczenia
e **goal function** r целевая функция; u цільова функція.; p funkcja celu
e **graphical output** r вывод графических данных; u виведення графічних даних; p dane obrazu wyjściowego
e **gray scale** r шкала яркостей; u шкала яскравостей; p skala jasności
e **grayscale picture** r полутоновое изображение; u напівтонове зображення; p półtonów obrazu
e **grid line** r линия координатной сетки; u лінія координатної сітки; p linii siatki

H

e **Hamming code** r код Хэмминга; u код Хеммінга; p kod Hamminga
e **Hermitian curve** r эрмитовая кривая; u ермітова крива; p krzywa Hermitian
e **hierarchical** r иерархический; u ієрархічний; p hierarchiczny
e **high-pass filter** r фильтр верхних частот; u фільтр верхніх частот; p częstotliwości filtru wyższego
hyperbolic equations r гиперболическое уравнение; u гіперболічне рівняння; p równanie hiperboliczne
e **homogeneous** r однородный; u однорідний; p jednorodny
e **HSB (hue, saturation, brightness)** r цвет, насыщенность, яркость; u колір, насиченість, яскравість; p jasność, nasycenie, barwy
e **homeomorphism** r гомеоморфизм; u гомеоморфізм; p homeomorfizmem

I

e **identity matrix** r единичная матрица; u одинична матриця; p tożsamość matrix
e **image acquisition** r получение изображений; u одержання зображень; p akwizycji obrazu
e **image amplifier** r усилитель изображений; u підсилювач зображень; p wzmacniacz obrazu
e **image processing** r обработка изображений; u обробка зображень; p obróbka obrazów
e **image spectrum** r спектр изображения;

u спектр зображення; p spektrum obrazu
e **implicit conversion** r неявное преобразование; u неявне перетворення; p niejawna konwersja
e **inaccessible value** r недоступное значение; u недоступне значення; p niedostępne wartość
e **inaccuracy** r неточность, погрешность; u неточність, похибка; p niedokładność, błąd
e **inch** r дюйм; u дюйм; p cal
e **incremental** r пошаговый; u покроковий; p krok po kroku
e **indiscrete** непрерывный; u неперервний; p nieprzerwany
e **inequality** r неравенство; u нерівність; p nierówność
e **infinite impulse response filter** r фильтр с бесконечной импульсной характеристикой; u фільтр з нескінченно імпульсною характеристикою; p filtr z nieskończoną impulsową charakterystyką
e **initial approximation** r начальное приближение; u початкове наближення; p początkowe przybliżenie
e **integer number** r целое число; u ціле число; p liczba całkowita
e **internal function** r внутренняя функция; u внутрішня функція; p wewnętrznych funkcji
e **interpolation** r интерполяция; u інтерполяція; p interpolacja
e **interpolation 'ahead'** r интерполяция «вперед»; u інтерполяція «вперед»; p interpolacja «naprzód»
e **interpolation 'back'** r интерполяция «назад»; u інтерполяція «назад»; p interpolacja «nazad»
e **inverse matrix** r обратная матрица; u обернена матриця; p macierz odwrotna
e **iterative algorithm** r итерационный алгоритм; u ітераційний алгоритм; p algorytm iteracyjny
e **isomorphism** r изоморфизм; u ізоморфізм; p izomorfizm

J

e **Jacobian** r якобиан, (функциональный) определитель Якоби; u якобіан; p âkobian
e **jitter** r дрожание, небольшие искажения сигнала; u тремтіння, незначні спотворення сигналу; p drganie
e **joint distribution** r совместное распределение; u спільний розподіл; p wspólna dystrybucja
e **juncture** r узловая точка; u вузлова точка; p punkt węzłowy
e **justification** r подтверждение; u підтвердження; p potwierdzenie

K

e **kernel** r ядро; u ядро; p jądro
e **key variable** r ключевая переменная; u ключова змінна; p kluczową zmienną
e **knot** r узел; u вузол; p węzeł
e **kink** r точка излома (кривой); точка зламу (кривої); punkt złomu

L

e **label** r метка; u мітка; p znak
e **Lagrangian** r лагранжиан, функция Лагранжа; u функція Лагранжа; p funkcja Lagrange
e **least square method** r метод наименьших квадратов; u метод найменших квадратів; p metoda najmniejszych kwadratów
e **limitation (truncation) error** r ошибка ограничения; u помилка обмеження; p pomyłka ograniczenia
e **linear equation system** r система линейных уравнений; u система лінійних рівнянь; p system liniowy równanie
e **linear interpolation** r линейная интерполяция; u лінійна інтерполяція; p interpolacja liniowa
e **lossless compression** r сжатие без потерь; u стиск без втрат; p kompresja bezstratna
e **low resolution mode** r графический режим с низким разрешением; u графічний режим з низькою роздільною здатністю; p trybie niskiej rozdzielczości
e **low-frequency cutoff** r нижняя граничная частота; u нижня гранична частота; p niższej częstotliwości kredytu
e **low-pass filter** r фильтр низких частот; u фільтр низьких частот; p filtr niskich częstotliwości

M

e **magnify** r увеличивать; u збільшувати; p wzrost

e **magnitude** r абсолютное значение; u абсолютне значення; p absolutne znaczenie

e **mathematical expectation** r математическое ожидание; u математичне сподівання; p matematyczne oczekiwanie

e **matrix multiplication** r перемножение матриц; u перемножування матриць; p mnożenie macierzy

e **matrix rank** r ранг матрицы; u ранг матриці; p ranking matrix

e **maximum likelihood** r максимальное правдоподобие; u максимальна правдоподібність; p największej wiarygodności

e **mean square approximation** r среднеквадратичное приближение; u середньоквадратичне наближення; p średnia kwadratowa zbliżenia

e **mean value** r среднее значение; u середнє значення; p średnia walor

e **method of 'shooting'** r метод «стрельбы»; u метод «стрільби»; p metoda «strelanie»

e **model adequacy** r адекватность модели; u адекватність моделі; p adekwatność modelu

e **model with distributed parameters** r модель с распределенными параметрами; u модель з розподіленими параметрами; p model o parametrach rozłożonych

e **multidimensional net** r многомерная сетка; u багатовимірна сітка; p wielowymiarowy siatka

e **multistep method** r многошаговый метод; u багатокроковий метод; p metoda bagatokrokovy

N

e **n-dimensional** r n-мерный; u n-вимірний; p n-wymiarowa

e **neighborhood** r окрестность; u окіл; p okolica

e **net** r сетка; u сітка; p siatka, sieć

e **node** r узел, узловая точка; u вузол, вузлова точка; węzeł

e **noise protection** r помехозащищенность; u завадозахищеність; p zavadozahyschenist

e **noise ratio** r коэффициент шума; u коефіцієнт шуму; p współczynnik szumów

e **non_linear regression mode** r нелинейная регрессионная модель; u нелінійна регресійна модель; p nieliniowy model regresji

e **non-stationary system** r нестационарная система; u нестаціонарна система; p niestacjonarnej systemie

e **nonsingular matrix** r невырожденная матрица; u невироджена матриця; p nieosobliwych matrix

e **number of iterations** r число итераций; u число ітерацій; p liczba iteracji

e **number variable** r числовая переменная; u числова змінна; p zmienna liczba

e **numerical approximation** r численная аппроксимация; u чисельна апроксимація; p numeryczna aproksymacja

e **numerical differentiation** r численное дифференцирование; u чисельне диференціювання; p numeryczne zróżnicowanie

e **numerical integration** r численное интегрирование; u чисельне інтегрування; p całkowanie numeryczne

e **normal distribution law** r нормальный закон распределения; u нормальний закон розподілу; p rozkład normalny

O

e **obtainable** r допустимый; u допустимий; p dopuszczalny

e **odd** r нечётный; u непарний; p nieparzysty

e **one-sweep method** r одношаговый метод; u однокроковий метод; p metoda jednoetapowa

e **operator scheme** r операторная схема; u операторна схема; p operator systemu

e **optimization techniques** r методы оптимизации; u методи оптимізації; p technik optymalizacji

e **order** r порядок; u порядок; p kolejność
e **order of magnitude** r порядок величины; u порядок величини; p rząd wielkości
e **orthogonal transformation** r ортогональное преобразование; u ортогональні перетворення; p ortogonalne przekształcenie
e **output signal** r выходной сигнал; u вихідний сигнал; p sygnał wyjściowy

P

e **parabolic equation** r параболическое уравнение; u параболічне рівняння; p równanie paraboliczne
e **parametric equation** r параметрическое уравнение; u параметричне рівняння; p równanie parametryczne
e **partial derivative** r частная производная; u частинна похідна; p pochodna cząstkowa
e **particularization** r детализация; u деталізація; p detalizowanie
e **pattern recognition** r распознавание образов; u розпізнавання образів; p rozpoznawanie wzorców
e **plural** r множество; u множина; p liczba mnoga
e **polynomial** r полином; u поліном; p wielomian
e **polynomial spline interpolation** r полиномиальная интерполяция сплайнами; u поліноміальна інтерполяція сплайнами; p splajny wielomianowe interpolacji
e **posterior probability** r апостериорная вероятность; u апостеріорна ймовірність; p posteriori prawdopodobieństwo
e **prior distribution** r априорное распределение; u апріорний розподіл; p przed dystrybucja
e **probability** r вероятность; u вірогідність; p prawdopodobieństwo
e **probability density** r плотность вероятности; u щільність вірогідності; p gęstość prawdopodobieństwa
e **probabilistic image models** r вероятностные модели изображений; u імовірнісні моделі зображень; p model probabilistyczny dla obrazów

Q

e **qualitative analysis** r качественный анализ; u якісний аналіз; p analiza jakościowa
e **quantization** r квантование; u квантування; p kwantyzacja

R

e **random process** r случайный процесс; u випадковий процес; p proces stochastyczny, proces losowy
e **random signal** r случайный сигнал; u випадковий сигнал; p sygnał losowy
e **rapid convergence** r быстрая сходимость; u швидка збіжність; p szybka konwergencja
e **recovery** r восстановление; u відновлення; p regeneracja
e **recurrence** r рекуррентное соотношение; u рекурентне співвідношення; p nawrót
e **recursion theorem** r теорема рекурсии; u теорема рекурсії; p twierdzenie rekursji
e **recursive filter** r рекурсивный фильтр; u рекурсивний фільтр; p filtr rekurencyjne
e **reduction without any loss** r сжатие без потерь; u стиск без втрат; p stisk bez vtrat
e **reference model** r эталонная модель; u еталонна модель; p etalon modelu
e **relative error** r относительная погрешность; u відносна похибка; p vidnosna pohibka
e **rigid task** r жесткая задача; u жорстка задача; p zadanie twarde
e **root** r корень; u корінь; p pierwiastek
e **rounding error** r погрешность округления; u похибка округлення; p zaokrąglanie pohibka
e **row** r ряд; u ряд; p rząd

S

e **sample collection** r выборка; u вибірка; p próbki
e **sample covariance function** r выборочная ковариационная функция; u вибіркова коваріаційна функція; p funkcja kowariancji próby
e **sampling error** r погрешность дискретизации; u похибка дискретизації; p pohibka diskretizatsii

e **scheduling** r построение графика; u побудова графіка; p budowa grafika
e **secant** r секущая; u січна; p sieczna
e **selfstarting** r самостартование; u самостартування; p samoczynny ruch
e **selfsimilar plural** r самоподобное множество; u самоподібна множина; p samopodobna mnoga
e **sequence** r последовательность; u послідовність; p ustalać kolejność
e **set** r группа; u група; p zbiór
e **share** r слой; u шар; p lemiesz
e **signal spectrum** r спектр сигнала; u спектр сигналу; p widmo sygnału
e **signal-to-noise ratio** r отношение сигнал/шум; u відношення сигнал/шум; p stosunek sygnał/szum
e **simple iteration** r простая итерация; u проста ітерація; p iteracja prosta
e **simulation** r имитационное моделирование; u імітаційне моделювання; p symulacja, modelowanie
e **simultaneous displacement** r одновременная подстановка; u одночасна підстановка; p jednoczesne przemieszczenie
e **spline interpolation** r сплайн-интерполяция; u сплайн-інтерполяція; p splajn interpolyatsiya
e **statistical processing** r статистическая обработка; u статистична обробка; p statystyczna obróbka
e **stochastic model** r стохастическая модель; u стохастична модель; p model stochastyczny
e **structured programming** r структурное программирование; u структурне програмування; p programowanie strukturalne
e **successive overhead relaxation** r последовательная верхняя релаксация; u послідовна верхня релаксація; p kolejna napowietrzna relaksacja
e **sweep method** r метод прогонки; u метод прогонки; p metoda przemiatania

T

e **tangent** r касательная; u дотична; p styczna
e **template** r шаблон; u шаблон; p szablon
e **transaction (digitization) error** r ошибка преобразования (дискретизации); u помилка перетворення (дискретизації); p pomyłka transakcji
e **transcendental equation** r трансцендентное уравнение; u трансцендентне рівняння; p transcendentalny równanie
e **transformer** r преобразователь; u перетворювач; p transformator
e **transient response** r переходная характеристика; u перехідна характеристика; p przejściowa charakterystyka
e **tridimensional space interpolation** r трехмерная интерполяция в пространстве; u тривимірна інтерполяція в просторі; p interpolacja przestrzeń trójwymiarowa

U

e **uncertainty** r неопределенность; u невизначеність; p niepewność
e **unidimensional net** r одномерная сетка; u одновимірна сітка; p siatka jednowymiarowa

V

e **vague** r неопределённый; u невизначений; p nieokreślony
e **validation** r проверка достоверности; u перевірка достовірності; sprawdzanie poprawności
e **variable** r переменная; u змінна; p zmienna
e **vicious position** r ложное положение; u хибне положення; p zjadliwe położenie

W

e **wavelet basis** r вейвлет-базис; u вейвлет-базис; p podstawą falkowa
e **wavelet decomposition** r вейвлет-разложение; u вейвлет-розкладання; p falkowa dekompozycja
e **wavelet compression** r вейвлет-сжатие; u вейвлет-стиск; p kompresja falkowa

Література

1. Gonzales R. C. Digital Image Processing Using MATLAB / R. C. Gonzales , R. E. Woods, S. Eddins. – Prentice Hall, Upper Saddle River, NJ, 2004. – 492 p.
2. Greenspan D. Introduction to Numerical Analysis and Applications / D. Greenspan. – Markham : Chicago, 1971. – 176 p.
3. Image Processing Toolbox For Use with Matlab, User's Guide. Version 3. – The Math Works Inc., 2004. – 775 p.
4. Kvyetnyy R. Basics of Modelling and Computational Methods / R. Kvyetnyy. – Вінниця : ВДТУ, 2007. – 147 с.
5. Методы компьютерной обработки изображений / Под ред. В. А. Сойфера – 2 изд. испр. – М. : ФИЗМАТЛИТ, 2003. – 784 с.
6. Абрамовиц М. Справочник по специальным функциям / Абрамовиц М. , Стиган И. – М. : Наука, 1979. – 486 с.
7. Амосов А. А. Вычислительные методы для инженеров : учеб. пособ. / Амосов А. А. , Дубинский Ю. А., Копченова Н. В. – М. : Высшая школа, 1994. – 554 с.
8. Анисимов Б. В. Распознавание и цифровая обработка изображений / Анисимов Б. В., Курганов В. Д., Злобин В. К. – М. : Высшая школа, 1983. – 468 с.
9. Аттетков А. В. Методы оптимизации : учеб. для вузов / Аттетков А. В. , Галкин С. В., Зарубин В. С. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2003. – 440 с.
10. Батищев Д. И. Методы оптимального проектирования / Батищев Д. И. – М. : Радио и связь, 1984. – 248 с.
11. Бахвалов Н. С. Численные методы / Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. – М. : Наука, 1987. – 630 с.
12. Васильков Ю. В. Компьютерные технологии вычислений в математическом моделировании : учеб. пособие / Ю. В. Васильков, Н. Н. Василькова. – М. : Финансы и статистика, 2002. – 256 с.
13. Верлань А. Ф. Интегральные уравнения: методы, алгоритмы, программы : справочное пособие / А. Ф. Верлань, В. С. Сизиков. – К. : Наукова думка, 1986. – 544 с.
14. Вержбицкий В. М. Численные методы (математический анализ и обыкновенные дифференциальные уравнения) : учеб. пособие для вузов / Вержбицкий В. М. – М. : Высш.шк., 2001. – 382 с.
15. Воеводин В. В. Матрицы и вычисления / Воеводин В. В. – М. : Наука, 1984. – 320 с.
16. Глинченко А. С. Цифровая обработка сигналов / Глинченко А. С. – Красноярск : Изд-во КГТУ, 2001. – 199с.

17. Глушков В. М. Основы безбумажной информатики / Глушков В. М. – М. : Наука, 1987. – 552 с.
18. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.
19. Горелик А. Л. Методы распознавания / А. Л. Горелик, В. А. Скрипкин. – М. : Высшая школа, 1984. – 208 с.
20. Грузман И. С. Цифровая обработка изображений в информационных системах / И. С. Грузман, В. С. Киричук. – Новосибирск : Изд-во НГТУ, 2002. – 352 с.
21. Дахнович А. А. Дискретные системы и цифровая обработка сигналов : учебное пособие / Дахнович А. А. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. – 100с.
22. Демидович Б. П. Основы вычислительной математики / Б. П. Демидович, И. А. Марон. – М. : Наука, 1970. – 664 с.
23. Дубовой В. М. Основи застосування ЕОМ у інженерній діяльності / В. М. Дубовой, Р. Н. Квєтний. – К. : ІСДО України, 1994. – 285 с.
24. Дубовой В. М. Програмування комп'ютеризованих систем управління та автоматики / В. М. Дубовой, Р. Н. Квєтний. – Вінниця : ВДТУ, 1997. – 208 с.
25. Дубовой В. М. Програмування персональних комп'ютерів систем управління / В. М. Дубовой, Р. Н. Квєтний. – Вінниця : ВДТУ, 1999.
26. Каллан Р. Основные концепции нейронных сетей / Каллан Р. – М. : Изд. дом «Вильямс», 2001. –288 с.
27. Канторович Л. В. Оптимальные решения в экономике / Л. В. Канторович, А. Б. Горстков – М. : Наука, 1972. – 335 с.
28. Квєтный Р. Н. Математическое моделирование в задачах проектирования средств автоматики и информационно-измерительной техники / Квєтний Р. Н. – К. : УМК ВО,1989. – 112 с.
29. Квєтный Р. Н. Информационная теория измерений: от модели к изделию / Р. Н. Квєтний, В. Т. Маликов. – М. : Знание, 1988. – 213 с.
30. Квєтний Р. Н. Інтервальні моделі перетворень сигналів в інформаційно-вимірювальних системах / Р. Н. Квєтний, О. Р. Бойко. – Вінниця : УНІВЕРСУМ-Вінниця, 2009. – 88с.
31. Різницеві методи та сплайни в задачах багатовимірної інтерполяції / [Квєтний Р. Н., Дементьєв В. Ю., Машницький М. О., Юдін О. О.]. – Вінниця : УНІВЕРСУМ-Вінниця, 2009. – 87 с.
32. Квєтний Р. Н. Методи фільтрації текстурованих зображень у задачах розпізнавання та класифікації / Р. Н. Квєтний, О. Ю. Софина. – Вінниця : УНІВЕРСУМ-Вінниця, 2011. – 119 с.
33. Коллатц Л. Функциональный анализ и вычислительная математика / Коллатц Л. – М. : Мир, 1969. – 448 с.
34. Корн Г. Справочник по математике для научных работников и инженеров / Г. Корн, Т. Корн. – М. : Наука, 1986. – 832 с.

35. Краскевич В. Е. Численные методы в инженерных исследованиях / Краскевич В. Е., Зеленский К. Х., Гречко В. И. – К. : Вища школа, 1986. – 264 с.
36. Ляшенко М. Я. Чисельні методи : підручник / М. Я. Ляшенко, М. С. Головань. – К. : Либідь, 1996. – 288 с.
37. Мак-Кракен Д. Численные методы и программирование на ФОРТРАНЕ / Д. Мак-Кракен, У. Дорн. – М. : Мир, 1977. – 584 с.
38. Маликов В. Т. Вычислительные методы и применение ЭВМ / В. Т. Маликов, Р. Н. Кветный. – К. : Вища школа, 1989. – 362 с.
39. Марчук Г. И. Введение в проекционно–сеточные методы / Г. И. Марчук, В. И. Агошков. – М. : Наука, 1982. – 264 с.
40. Рабинер Л. Теория и применение цифровой обработки сигналов / Л. Рабинер, Б. Гоулд ; пер. с англ. – М. : Мир, 1979. – 578 с.
41. Ракитский Ю. В. Численные методы решения жестких систем / Ракитский Ю. В., Устинов С. М., Черноруцкий И. Г. – М. : Наука, 1979. – 208 с.
42. Рудаков П. И. Обработка сигналов и изображений. Matlab 5.x / П. И. Рудаков, И. В. Сафонов. – М. : ДИАЛОГ-МИФИ, 2000. – 416 с.
43. Саати Т. Целочисленные методы оптимизации и связанные с ними экстремальные проблемы / Саати Т. – М. : Мир, 1973. – 302 с.
44. Самарский А. А. Введение в численные методы / Самарский А. А. – М. : Наука, 1987. – 234 с.
45. Самарский А. А. Теория разностных схем / Самарский А. А. – М. : Наука, 1977. – 400 с.
46. Скурихин В. Н. Математическое моделирование / Скурихин В. Н., Шифрин В. Б., Дубровский В. В. – К. : Техніка, 1983. – 270 с.
47. Трусов П. В. Введение в математическое моделирование : учеб. пособие / Трусов П. В. – М. : Логос, 2005. – 440 с.
48. Фисенко В. Т. Компьютерная обработка и распознавание изображений : учеб. пособие / В. Т. Фисенко, Т. Ю. Фисенко. – СПб. : СпбГУ ИТМО, 2008. – 192 с.
49. Фельдман Л. П. Чисельні методи в інформатиці : підручник / [Фельдман Л. П., Згуровський М. З., Петренко А. І., Дмитрієва О. А.]. – К. : Вид. група BHV, 2006. – 480 с.
50. Форсайт Дж. Машинные методы математических вычислений / Дж. Форсайт, М. Малькольм, К. Моулер. – М. : Мир, 1980. – 279 с.
51. Шуп Т. Решение инженерных задач на ЭВМ / Шуп Т. – М. : Мир, 1982. – 238 с.
52. Чабан В. Чисельні методи / Чабан В. – Львів : Вид. Нац. ун-ту "Львівська політехніка", 2001. – 186 с.
53. Яковлев А. Н. Введение в вейвлет-преобразования / Яковлев А. Н. – Новосибирск : Изд-во НГТУ, 2003. – 104 с.

Навчальне видання

Квєтний Роман Наумович
Богач Ілона Віталіївна
Бойко Олексій Романович
Софина Ольга Юріївна
Шушура Олексій Миколайович

КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ СИСТЕМ ТА ПРОЦЕСІВ. МЕТОДИ ОБЧИСЛЕНЬ
Частина 2

Навчальний посібник

Редактор В. Дружиніна
Оригінал-макет підготовлено О. Софиною

Підписано до друку 28.02.2013 р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. др. арк. 13,57.
Наклад 300 (1-й запуск – 100) прим. Зам № 2013-063.

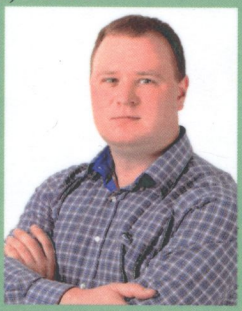
Вінницький національний технічний університет,
комп'ютерний інформаційно-видавничий центр.
21021, м. Вінниця, Хмельницьке шосе, 95.
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-85-32.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95.
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-38.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Квєтний Роман Наумович, д.т.н., професор
завідувач кафедри автоматики та інформаційно-вимірювальної техніки Вінницького національного технічного університету

Богач Ілона Віталіївна, к.т.н.,
доцент кафедри автоматики та інформаційно-вимірювальної техніки, ВНТУ

Софина Ольга Юріївна, к.т.н.,
старший викладач кафедри автоматики та інформаційно-вимірювальної техніки, ВНТУ



Бойко Олексій Романович, к.т.н.,
доцент кафедри автоматики та інформаційно-вимірювальної техніки, ВНТУ

Шушура Олексій Миколайович, к.т.н.,
доцент кафедри системного аналізу і моделювання Донецького національного технічного університету