|  |  |
| --- | --- |
| Міністерство освіти та науки Українинаціональний університет «Одеська політехніка»інститут штучного інтелекту та робототехніки **Кафедра програмних та комп'ютерно-інтегрованих технологій**  Методичні вказівки з дисципліні математичне забезпечення кіск (Теоретична частина)  Для студентів інституту штучного інтелекту та робототехніки  Другий (магістерський) рівень вищої освіти  Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології  Освітньо-професійна програма: Комп'ютерні технології автоматизації;  Кількість рік/кредитів ЄКТС за навчальним планом: 135/4,5 | |
|  | Одобрено на засіданні кафедри ПКІТ протокол №7 від 26.01.2022 р. |
| **Одеса 2022** | |

Методичні вказівки з дисципліни «Математичне забезпечення КІСК» (Теоретична частина) для студ. спец. 151 – «Автоматизація та комп'ютерно-інтегровані технології» ден. та заоч. форм навч./уклад.: В.О. Давидов – Одеса: ОП, 2022. – 68 с.

|  |  |
| --- | --- |
| Укладачі: | **В.О. Давидів**, Канд. техн. наук |
|  |  |
|  |  |
|  |  |
|  |  |

Зміст

[1 Обчислення з плаваючою точкою 5](#_Toc97021596)

[1.1 Числа з плаваючою точкою 5](#_Toc97021597)

[1.2 Обчислення машинного епсілон 7](#_Toc97021598)

[1.3 Приклад помилок заокруглення 7](#_Toc97021599)

[1.4 Нестійкість деяких алгоритмів 9](#_Toc97021600)

[1.5 Чутливість деяких завдань 10](#_Toc97021601)

[1.6 Розв'язання квадратних рівнянь 13](#_Toc97021602)

[2 Формування випадкових сигналів (перешкод) 16](#_Toc97021603)

[2.1 Формування чисел із рівномірним розподілом 16](#_Toc97021604)

[2.2 Формування чисел із нормальним розподілом 17](#_Toc97021605)

[2.2.1 Метод Ziggurat 20](#_Toc97021606)

[2.2.2 Метод підсумовування 22](#_Toc97021607)

[2.3 Формування чисел з експонентним розподілом 22](#_Toc97021608)

[2.4 Формування чисел із гамма-розподілом 24](#_Toc97021609)

[2.4.1 Алгоритм GA 24](#_Toc97021610)

[2.4.2 Алгоритм GS 25](#_Toc97021611)

[2.4.3 Алгоритм GF 25](#_Toc97021612)

[2.4.4 Алгоритм GO 26](#_Toc97021613)

[2.5 Формування чисел із розподілом Коші 27](#_Toc97021614)

[2.6 Формування чисел із розподілом Лапласа 27](#_Toc97021615)

[2.7 Формування чисел з розподілом Леві 28](#_Toc97021616)

[2.8 Формування чисел із розподілом хі-квадрат 29](#_Toc97021617)

[2.9 Формування чисел із логнормальним розподілом 29](#_Toc97021618)

[2.10 Формування чисел із логістичним розподілом 30](#_Toc97021619)

[3 Математична модель надійності обладнання 32](#_Toc97021620)

[4 Математична модель ефективності теплотехнічного обладнання 39](#_Toc97021621)

[5 Методи однокритеріальної оптимізації 41](#_Toc97021622)

[5.1 Загальні відомості про градієнтні методи 41](#_Toc97021623)

[5.2 Метод градієнтного спуску 42](#_Toc97021624)

[6 Багатокритеріальна оптимізація 48](#_Toc97021625)

[6.1 Проблеми та класифікація методів вирішення задач багатокритеріальної оптимізації 49](#_Toc97021626)

[6.2 Метод адитивної згортки 49](#_Toc97021627)

[6.3 Метод мультиплікативної згортки 51](#_Toc97021628)

[6.4 Метод цільового програмування 52](#_Toc97021629)

[6.5 Метод аналізу ієрархій 53](#_Toc97021630)

[7 Апроксимація 60](#_Toc97021631)

[7.1 Апроксимація лінійною функцією 60](#_Toc97021632)

[7.2 Апроксимація іншими залежностями 61](#_Toc97021633)

[8 Інтерполяція 63](#_Toc97021634)

[8.1 Канонічний поліном 66](#_Toc97021635)

[8.2 Лінійна інтерполяція 66](#_Toc97021636)

[8.3 Інтерполяція кубічними сплайнами 67](#_Toc97021637)

# 1 Обчислення з плаваючою точкою

## 1.1 Числа з плаваючою точкою

Система дійсних чисел з її невдалою назвою настільки міцно утвердилася в фундаменті математичного аналізу, що легко забути, що всі дійсні числа неможливо уявити у світі кінцевих обчислювальних машин. Однак, хоч би як спрощувала аналіз система дійсних чисел, практичні обчислення змушені обходитися без неї.

Багато різних методів було запропоновано для апроксимації дійсних чисел у вигляді кінцевих машинних уявлень. Метод, прийнятий нині майже всіх машинах, — це числа з плаваючою точкою. Безліч F чисел із плаваючою точкою характеризується чотирма параметрами: основою β, точністю t та інтервалом показників [L, U]. Кожне число х з плаваючою точкою, що належить F, має значення



де цілі числа d1,..., dt задовольняють нерівності

0 ≤ di ≤ β -1 (i = l, ..., t)

і L ≤ e ≤ U. Якщо для кожного ненульового х із F справедливо d1 ≠ 0, то плаваюча система F називається нормалізованою. Ціле число е називається показником, а число f = (d1/β+…+ dt/βt)

*- Дробовою частиною.*Зазвичай ціле число βt∙f зберігається за тією чи іншою схемою уявлення, прийнятої цілих чисел, на кшталт величини зі знаком, доповнення до одиниці чи доповнення до двох.

Дійсна машинна реалізація плаваючих уявлень може відрізнятися в деталях від ідеальної, що обговорюється тут, проте відмінності несуттєві, і їх майже завжди можна ігнорувати, говорячи про основні проблеми округлення.

Наступна таблиця дає деякі приклади систем із плаваючою точкою. Розмір β1-t є оцінкою відносної точності арифметики. Тут не наводиться точного значення машинного эпсилон, оскільки воно залежить від деяких тонких деталей, таких як спосіб округлення.

| Обчислювальна машина | β | *t* | *L* | *U* | β1-t |
| --- | --- | --- | --- | --- | --- |
| Uni vac 1108 | 2 | 27 | -128 | 127 | 1.49× 10-8 |
| Honeywell 6000 | 2 | 27 | -128 | 127 | 1.49 × 10-8 |
| PDP-11 | 2 | 24 | -128 | 127 | 1.19 × 10-7 |
| Control Data 6600 | 2 | 48 | -976 | 1070 | 7.11 × 10-15 |
| Cray-1 | 2 | 48 | -16384 | 8191 | 7.11 × 10-15 |
| llliac-IV | 2 | 48 | -16384 | 16383 | 7.11 × 10-15 |
| Сетунь | 3 | 18 | -121 | 121 | 7.74 × 10-9 |
| Burroughs B55OO | 8 | 13 | -51 | 77 | 1.46 × 10-11 |
| Hewlett Packard HP-45 | 10 | 10 | -98 | 100 | 1.00 × 10-9 |
| Texas Instruments SR-5x | 10 | 12 | -98 | 100 | 1.00 × 10-11 |
| IBM 360 та 370 | 16 | 6 | -64 | 63 | 9.54 × 10-7 |
| IBM 360 та 370 | 16 | 14 | -64 | 63 | 2.22 × 10-16 |
| Telefunken TR440 | 16 |  | -127 | 127 | 5.84 × 10-11 |
| Maniac 11 | 65536 |  | -7 | 7 | 7.25 × 10-9 |

Якщо число t є цілим, це означає, що β = 2k і для двійкового уявлення дробової частини відводиться k∙t біт.

На деяких обчислювальних машинах використовується кілька систем чисел із плаваючою точкою. Наприклад, IBM 360 має дві системи з основою 16, вказані в таблиці. Ці дві різні системи називаються звичайною точністю та подвоєною точністю.

Багато F не є континуумом або навіть нескінченним безліччю. У ньому рівно 2(β-1) βt-1(U-L + 1) + 1 чисел. Вони розташовані нерівномірно; рівномірність розташування має місце лише за фіксованого показника. На рис. 1 показано 33-точкове безліч F для невеликої ілюстративної системи з параметрами β = 2, t = 3, L = -1, U = 2.

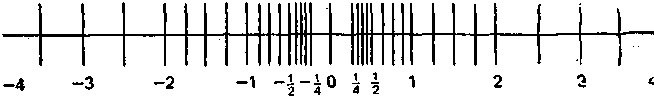


Рис. 1. Система чисел з плаваючою точкою для β = 2, t = 3, L = -1, U = 2

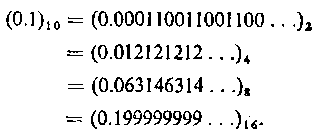
Оскільки F - кінцева множина, неможливо скільки-небудь детально відобразити континуум дійсних чисел. Більш того, дійсні числа з модулем, більшим за максимальний елемент з F, не можуть бути відображені взагалі. З ряду причин те саме вірно щодо ненульових дійсних чисел, менших за абсолютною величиною порівняно з найменшим позитивним числом з F. Нарешті, кожне число з F повинно представляти цілий інтервал дійсних чисел. Якщо х — дійсне число, що не виходить за межі множини F, то позначатимемо через fl(х) елемент з F, найближчий до х. (Зауважимо, що якщо х рівновіддалено від двох чисел з F, то fl(x) може приймати будь-яке з двох сусідніх з х значень.) Легко довести, що для числа х, що знаходиться в межах множини F. справедлива нерівність



Повчальний приклад дає десяткове число 0.1, яке часто береться як величина кроку в багатьох алгоритмах, що обговорюються надалі. Чи становитимуть десять кроків довжини 0.1 те саме, що один крок довжини 1.0? Ні, це не так у плаваючій системі, у якої β є ступенем 2, оскільки 1/10 не має кінцевого розкладання за ступенями 1/2. Справді,



Використовуючи індекси для позначення основи системи, можемо написати



У дробах правої частини необхідно обмежитися t розрядами. Коли складається десять таких дробів, результат не буде точною одиницею.

На множині F, що є моделлю системи дійсних чисел, визначені арифметичні операції відповідно до того, як вони виконуються цифровою обчислювальною машиною. Припустимо, що х і у числа з плаваючою точкою. Тоді нормальна сума х+у часто вже не належить F. Наприклад, візьмемо 33-точкову систему, зображену на рис. 1, і покладемо х = 5/4 та y = 3/8. Таким чином, операція додавання, наприклад, сама повинна моделюватися в машині за допомогою наближення, званого плаваючим додаванням. Якщоі число х+у не виходить за межі множини F, то ідеалом було б виконання рівності



депозначає плаваюче додавання. У більшості обчислювальних машин цей ідеал досягається або майже досягається для всіх таких х і у. Тому в нашій іграшковій 33-точковій множині F слід очікувати, що 5/43/8 і 3/2 або 7/4. Різниця між xy і х+у (для х, у F) є помилка округлення, досконала при плаваючому додаванні. Аналогічні властивості правильні для плаваючих операцій віднімання, множення та розподілу.

Причина, за якою число 5/4 + 3/8 не належить 33-точковій множині F, пов'язана з розташуванням елементів F. З іншого боку, сума типу 7/2 + 7/2 не належить F тому, що 7 більше максимального елемента з F. Спроба утворити таку суму викличе на більшості машин сигнал переповнення, після чого обчислення зазвичай перериваються, оскільки неможливо дати розумне наближення для чисел, що виходять за межі множини F.

У той час як багато сум х + у (для х, у з F) самі знаходяться в F, дуже рідко звичайне твір х-у належить F, оскільки, як правило, воно записується за допомогою 2t або 2t-1 значущих цифр. Крім того, переповнення набагато ймовірніше при множенні. Нарешті, при плаваючому множенні можливе виникнення машинного нуля, коли два ненульові числа х і у мають ненульовий добуток, менший за абсолютною величиною найменшого ненульового елемента з F. (Виникнення машинного нуля можливе і при складанні, хоча це дуже рідко). Таким чином, змодельована операція множення передбачає округлення ще частіше, ніж плаваюча додавання.

Операції плаваючого додавання та множення комутативні, але з асоціативні, і дистрибутивний закон їм також не выполняется. Оскільки ці закони алгебри мають фундаментальне значення для математичного аналізу, аналіз плаваючих обчислень складний. Існують методи аналізу помилок округлення, які обходять деякі з цих основних труднощів, і за допомогою таких методів було проаналізовано ряд чисельних алгоритмів, проте це питання виходить за межі цієї книги.

## 1.2 Обчислення машинного епсілон

Точність плаваючої арифметики можна характеризувати за допомогою машинного эпсилон, тобто найменшого числа з плаваючою точкою е, такого, що



Хоча це визначення виділяє в якості машинного эпсилон однину плаваючою системи, не так вже й важливо використовувати його точне значення. Воно зазвичай використовується в програмах таким чином, що може відрізнятися від свого точного визначення декількома ступенями 2, і це не викликає серйозних труднощів.

Існує кілька методів обчислення значення (або наближеного значення) є. Таким чином, програма може визначити точність машини, на якій вона виконується, під час виконання. Метод, яким ми обчислюємо наближення, що відрізняється від найбільшого множником 2, ілюструється наступним сегментом фортранної програми;

EPS = 1.

10 EPS = 0.5 \* EPS

EPSP1 = EPS+1.

IF ( EPSPl .GT.1. ) GO TO 10

## 1.3 Приклад помилок заокруглення

Важливою функцією аналізу є показова функція ех. Оскільки вона використовується дуже часто, важливо мати можливість обчислювати їх машинною програмою для будь-якого числа з плаваючою точкою х. Існує безліч методів, на яких вона може бути заснована, і більшість обчислювальних систем наукового призначення мають таку програму. Але припустимо, що на нашій машині такої програми немає, і запитаємо себе, як ми її склали б. Ця ситуація може виникнути для більш складної трансцендентної функції або нової обчислювальної машини.

Згадаймо, що для будь-якого дійсного (або навіть комплексного) значення х число ех може бути представлене як сума схожого нескінченного ряду



Спробуємо скористатися цією поряд для обчислення ех. Припустимо, що плаваюча система характеризується параметрами β = 10 і t=5. Застосуємо ряд для х = -5.5. Ось числа, які ми отримаємо:

*е*-5.5= 1.0000

-5.5000

+15.125

-27.730

+38.129

-41.942

+38.446

-30.208

+20.768

-12.692

+6.9803

-3.4902

+1.5997

.

.

.

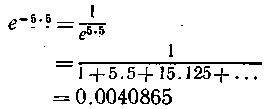
\_\_\_\_\_\_\_\_\_\_

+ 0.0026363

Ми обмежуємося під сумування 25 членами, оскільки наступні доданки вже не змінюють суму. Чи задовільну відповідь отримано? Істинний результат є*‑*5.5= 0.00408677, отже обчислення з допомогою низки призвело до відповіді, який має вірних цифр.

Що було неправильно? Зауважимо, деякі члени, і отже проміжні суми, на кілька порядків більше кінцевої відповіді. При цьому такі доданки, як 38.129, вже містять помилку округлення, майже рівну за величиною остаточного результату. Крім того, чотири старші (тобто найбільш значущі) розряди кожного з восьми членів, що перевищують 10 за модулем, були втрачені. Ці вісім членів слід брати з десятьма значущими цифрами, щоб отримати шість значущих цифр у відповіді. Більше того, знадобилася б ще одинадцята цифра, щоб можна було сподіватися отримати вірну шосту цифру суми, що обчислюється. Це іноді називається катастрофічною втратою вірних знаків, воно часто зустрічається в погано продуманих обчисленнях. Важливо, однак, розуміти, що така втрата знаків не є причиною помилки у відповіді;

Хоча іноді можна утримувати в обчисленнях більшу кількість значущих цифр з метою уникнути катастрофічної втрати знаків, це обходиться дорого як за часом виконання, так і по пам'яті, і часто вимагає спеціальних програмних засобів. Для завдання, що розглядається, є набагато кращі ліки: саме, обчислити суму для х = 5.5 і потім взяти зворотне число:



(У нашій п'ятизначній десятковій арифметиці). При такому способі обчислення помилка знижується до 0.007%.

Один із важливих висновків, що наведені з цього прикладу, — те, що навіть коротка послідовність обчислень може бути пов'язана із серйозними помилками округлення.

Зауважте, наскільки гіршим було б завдання, якби ми хотіли обчислити ех для х= –100.

Реальні машинні алгоритми для обчислення ех починаються з розкладання х на дійсну та дробову частини



*т -*ціле, 0 ≤ f ≤ 1. Далі, використовуючи властивості показової функції, отримуємо



Або

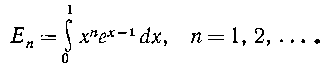


Кожна з величин, взятих у квадратні дужки, передбачає обчислення еy для у з інтервалу 0 ≤ y < 1. Обмеження на такий малий інтервал полегшує обчислення ey. Потім результат може бути помножений на ет (яке обчислюється просто) або зведений у ступінь т.

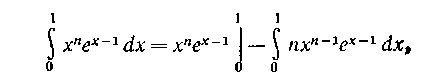
## 1.4 Нестійкість деяких алгоритмів

Завдання обчислення е-5.5, яке обговорювалося у попередньому параграфі, показує, як погано продуманий алгоритм може призвести до незадовільного результату навіть для цілком добре поставленого завдання. Проблема була подолана у вигляді зміни алгоритму. Для деяких завдань «хороші» відповіді не можна отримати жодним алгоритмом, тому що завдання чутливе до малих помилок, допущених при поданні даних та в арифметиці. Два приклади таких завдань наведено у наступному розділі. Важливо розрізняти ці два класи пасток, оскільки нестійкі алгоритми та чутливі завдання зустрічаються майже у всіх галузях обчислювальної математики. Однак, якщо ви знаєте їх симптоми, то діагностувати ці завдання вже досить просто. У цьому параграфі ми обговоримо наочніший приклад нестійкого алгоритму.

Припустимо, що хочемо обчислити інтеграли



Інтегруючи частинами, отримаємо



або



де Еn = 1/е. Нехай β = 10 та t = 6; ми можемо використовувати це рекурентне співвідношення для обчислення наближень до дев'яти перших значень Еп:

*E1*≈ 0.367879, *Е6*≈ 0.127120,

*Е2*≈ 0.264242, *E7*≈ 0.110160,

*Е3*≈ 0.207274, *E8*≈ 0.118720,

*Е4*≈ 0.170904, *Е9*≈ -0.0684800.

*Е5*≈ 0.145480,

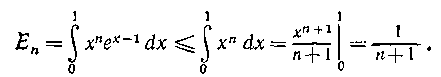
Хоча підінтегральний вираз х9еx-1 позитивно по всьому інтервалі (0,1), обчислене нами значення для Е9 негативно.

Що викликало таку велику помилку? Зауважимо, що єдина помилка округлення, зроблена в наведених обчисленнях, - це помилка в Е1, коли 1/е округляється до шести цифр. Оскільки рекурентна формула, отримана інтегруванням частинами, точна для дійсної арифметики, помилка в Е9 повністю зобов'язана помилці округлення, зроблена в Е1. Щоб зрозуміти, як помилка в Е1 приблизно дорівнює 4.412 × 10-7, стає настільки великою, зауважимо, що вона множиться на -2 при обчисленні Е2, потім помилка в Е2 множиться на -3 при обчисленні Е3 і т.д. помилка в Е9 є точно помилка в Е1, помножена на (-2) (-3) ... ... (-9) = 9! або приблизно 9! × 4.412 × 10-7 ≈ 0.1601. Це величезне збільшення помилки, що міститься у вихідних даних завдання, є результатом обраного алгоритму. Справжнє значення Е9 (з трьома цифрами) є – 0.

Як можна було б обрати інший алгоритм, який не має такої нестійкості? Якщо ми перепишемо рекурентне співвідношення у вигляді



то тепер на кожному кроці обчислення помилка в Еп множиться на множник 1/n. Таким чином, якщо ми почнемо значення для деякого Еп при n >> 1 і будемо обчислювати в зворотному напрямку, то будь-яка початкова помилка або проміжні помилки округлень будуть зменшуватися на кожному кроці. Це називається стійким алгоритмом. Щоб знайти початкове значення, зауважимо, що



Отже, Еп прагне нуля, коли п прагне ∞. Наприклад, якщо ми апроксимуємо нулем число Е20 і візьмемо цей нуль стартовим значенням, то зробимо початкову помилку, що не перевищує 1/21. Ця помилка множиться на 1/20 при обчисленні Е19, тому помилка в Е19 не перевищує (1/20)×(1/21) ≈ 0.0024. На час обчислення Е15 початкова помилка зменшиться до величини, меншої 4×10-8, що у своє чергу менше єдиної помилки округлення. Проводячи ці обчислення, отримуємо

Е20 ≈ 0.0, Е14 ≈ 0.0627322,

Е19 ≈ 0.0500000, Е13 ≈ 0.0669477,

Е18 ≈ 0.0500000, Е12 ≈ 0.0717733,

Е17 ≈ 0.0527778, Е11 ≈ 0.0773523,

Е16 ≈ 0.0557190, Е10 ≈ 0.0838771,

Е15 ≈ 0.0590176, Е9 ≈ 0.0916123.

Коли ми дійдемо до Е15, початкова помилка в Е20 вже зовсім пригнічена стійкістю алгоритму та обчислені значення від Е15 до Е9 точні у всіх шести значних цифрах, з точністю до можливої ​​помилки округлення в останній цифрі.

## 1.5 Чутливість деяких завдань

Ми покажемо тепер, що деякі обчислювальні завдання напрочуд чутливі до зміни даних. Цей аспект чисельного аналізу залежить від плаваючої системи чи обраного алгоритму.

Завдання знаходження коріння полінома є стандартним обчислювальним завданням; часто вона дуже чутлива до зміни даних. Наприклад, розглянемо квадратичний поліном з майже кратним корінням на кшталт



Коріння цього рівняння є 2±10-3. Однак зміна вільного члена на 10-6 може викликати зміну в корінні, що дорівнює 10-3. Цей тип нестійкості ще більш виражений у поліномів вищого ступеня.

Однак нестійкість можна спостерігати не тільки у поліномів з майже кратним корінням. Наступний приклад належить Вілкінсону (1963). Нехай



Коріння р(х) суть 1,2, …, 19, 20 і добре розділені. Цей приклад виник на тлі плаваючої системи з параметрами ? Припустимо, що тільки одного з двадцяти коефіцієнтів робиться помилка в молодшому двійковому розряді. Саме, припустимо, що коефіцієнт при х19 змінюється

з -210 на -210 +2-23. Який вплив це мала зміна! зробить на корені полінома?

Щоб відповісти на це запитання, Вілкінсон ретельно обчислив (при β = 2, t = 90) корені рівняння р (х) +2-23х19 = 0. Наводимо їх значення, правильно заокруглені до зазначеного числа цифр:

1.00000 0000 10.09526 6145 ± 0.64350 0904i

2.00000 0000 11.79363 3881 ± 1.65232 9728i

3.00000 0000 13.99235 8137 ± 2.51883 0070i

4.00000 0000 16.73073 7466 ± 2.81262 4894i

4.99999 9928 19.50243 9400 ± 1.94033 0347i

6.00000 6944

6.99969 7234

8.00726 7603

8.91725 0249

20.84690 8101

Зауважимо, що мала зміна в коефіцієнті –210 мало наслідком те, що десять коренів стали комплексними і що з них відсунулися від дійсної осі більш ніж на 2.81. Звичайно, повна запис р(х) у машині зажадала б значно більше заокруглень, а обчислення коренів пов'язане з ще більшими помилками.

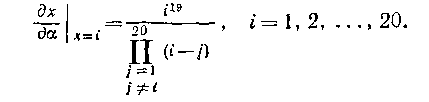
Наведена вище таблиця коренів була результатом дуже точних обчислень і постраждала скільки-небудь помітно від помилок округлення. Причина, через яку це коріння так сильно змінилося, лежить не в помилках округлення і не пов'язана з вибором алгоритму обчислення; суть у чутливості самої задачі. Легко провести аналіз того, що сталося. Можна записати поліном у вигляді



і потім знайти приватну похідну для кожного кореня полінома р(х). Це робиться диференціюванням рівняння р(х, α) = 0 по α:



Обчислюючи цей вираз для кожного кореня, отримуємо



Ці числа дають пряму міру чутливості кожного з коренів зміни коефіцієнта а. Наводимо їх значення:

|  |  |
| --- | --- |
| Корінь | *дх/так|х=i* |
| 1 | -8.2 × 10-18 |
| 2 | 8.2 × 10-11 |
| 3 | -1.6 × 10-6 |
| 4 | 2.2×10-3 |
| 5 | -6.1 × 10-1 |
| 6 | 5.8 × 101 |
| 7 | -2.5 × 103 |
| 8 | 6.0 × 104 |
| *9* | -8.3 × 105 |
| 10 | 7.6 × 106 |
| 11 | -4.6 × 107 |
| 12 | 2.0 × 108 |
| 13 | -6.1 × 108 |
| 14 | 1.3 × 109 |
| 15 | -2.1 × 109 |
| 16 | 2.4 × 109 |
| 17 | -1.9 × 109 |
| 18 | 1.0 × 109 |
| 19 | -3.1 × 108 |
| 20 | 4.3 × 107 |

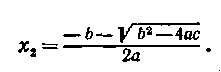
## 1.6 Розв'язання квадратних рівнянь

Є знаменитий алгоритм для вирішення квадратних рівнянь, що міститься в наступній математичній теоремі:

Теорема. Якщо а, b, с — дійсні числа і а≠0, то рівняння ах2+bх+с=0 має два рішення, саме



і



Подивимося, як ці формули, будучи застосовані «в лоб» для обчислення х1 і х2. На цей раз ми візьмемо плаваючу систему з β = 10, t = 8, L = -50, U = 50; вона точніша за багато широко використовуваних машинних систем.

*Випадок 1.*а = 1, b = -105, С = 1.

Точне коріння відповідного квадратного рівняння, правильно заокруглене до 11 значущих десяткових цифр, суть

*х*1= 99999.999990,

*х*2= 0.000010000000001.

Якщо ми скористаємося формулами теореми, то обчислимо

*х*1=100000.00 (дуже добре),

*х*2= 0 (абсолютно неправильно).

(Радимо читачеві перевірити, чи розуміє він, як х2 стає банкрутом у цьому плаваючому обчисленні.)

Знову, цього разу при обчисленні х2 ми постраждали від катастрофічної втрати вірних знаків. Існують різні альтернативні способи обчислення коренів квадратичного полінома, які уникають подібної втрати. Одним з таких способів є використання знака b для певна, яка формула тягне меншу втрату знаків, а потім обчислення одного з коренів за цією формулою. Таким чином,



Оскільки з рівності ах2+bх+с=a(x–х1) (х–х2) випливає aх1х2=с, то другий корінь можна обчислити за формулою



Для даних випадку 1 такий спосіб дає х1=100000.00, х2=1.0000000/100000.00=0.000010000000; обидва результати цілком прийнятні.

Зараз ми хотіли б запропонувати наступний критерій для реалізації машинного алгоритму розв'язання квадратних рівнянь. Він формулюється тут досить вільно, проте у статті Форсайта (1969) можна знайти суворе формулювання.

Ми скажемо, що комплексне число z знаходиться строго в межах множини F. якщо z=0, або



Це означає, що дійсна та уявна частини z знаходяться строго всередині області чисел, які можуть бути добре наближені елементами F. Досить довільний множник β2 залучений для того, щоб забезпечити деякий запас надійності.

Припустимо, що всі три числа а, b, з знаходяться строго в межах F. Тоді вони повинні бути допустимі як вхідні дані для алгоритму розв'язання квадратних рівнянь. Якщо а=b=с=0, то алгоритм повинен закінчувати свою роботу повідомленням, що це комплексні числа задовольняють рівнянню ах2+bх+с=0. Якщо а=b=0 і с≠0, то на виході алгоритму має бути інформація про те, що жодна комплексна кількість не задовольняє цього рівняння.

Інакше нехай z1 і z2 — точні корені рівняння, занумеровані отже |z1|≤|z2|. (Якщо a=0, покладемо z2=∞.) У всіх випадках, коли z1 знаходиться строго в межах множини F, алгоритм повинен знаходити хороше наближення до z1; під цим розуміється вимога, щоб наближення відрізнялося від z1 лише, скажімо, однією одиницю передостаннього знака кореня.

Те саме має виконуватися для z2.

Якщо один або два корені не знаходяться строго в межах множини F, то має видаватися відповідне повідомлення і той корінь, який лежить строго в межах F (якщо є), повинен бути визначений із зазначеною точністю.

Цим завершується вільний опис бажаного виконання алгоритму на вирішення квадратних рівнянь. Повернемося тепер до розгляду деяких типових рівнянь, щоб переглянути, як працюють для них квадратичні формули.

*Випадок 2*a = 6, b = 5, з = -4.

Тут немає жодних труднощів при обчисленні х1 = 0.50000000 і х2 = -1.3333333 або наближення до цих значень, хоч би якою формулою користуватися.

*Випадок 3.*а=6×1030, b=5×l030, з=–4×1030.

Оскільки, з точністю до множника 1030 це ті ж коефіцієнти, що і у випадку 2, то коріння не змінилися. Однак, застосування формул для х1 і х2, викликає переповнення, так як b2, будучи більше 1050, знаходиться за межами F. Швидше за все ці рівномірно великі значення для |а|, |b|, |с| зможуть бути виявлені до входу в алгоритм, і всі три числа будуть поділені на якийсь масштабуючий множник типу 1030, після чого завдання зведеться до випадку 2.

*Випадок 4.*a = 10-30, b = -1030, з = 1030.

Тут х1 приблизно дорівнює 1, у той час як х2, приблизно дорівнює 1060. Отже, наш алгоритм повинен визначити дуже точно, незважаючи на те, що х2 знаходиться за межами F. Очевидно, що будь-яка спроба досягти приблизної рівності модулів коефіцієнтів простим розподілом їх на одне і те ж число приречене на невдачу і як така може спричинити переповнення чи поява машинного нуля. Це рівняння є суворим випробуванням для будь-якої програми, що вирішує квадратні рівняння. |

Читач може подумати, що квадратне рівняння, у якого один корінь знаходиться в межах F, а інший поза ними, є штучний приклад, що не має практичного значення. У такому разі він помиляється. У багатьох ітераційних алгоритмах, що містять підпрограму розв'язання квадратних рівнянь, квадратичні багаточлени мають вироджену поведінку, що характеризується тим, що а→0 у разі збіжності алгоритму. Приклад такої ситуації — метод, запропонований у статті Мюллера (1956), знаходження нулів гладкої функції загального виду.

*Випадок 5.*а = 1.0000000, b = -4.0000000, з = 3.9999999.

Тут корінням є х1 = 1.999683772, х2 = 2.000316228. Однак застосування квадратичних формул дає

*х*1=*х*2=2.0000000, |

і лише перші чотири знаки вірні. Ці наближення далеко ще не задовольняють висунутому вище критерію, проте проти іншими прикладами труднощі тут зовсім інша.

Даний квадратичний поліном відповідає квадратному рівнянню (х-2) 2 = е, де е = 0.0000001. Як мовилося раніше у попередньому параграфі, це випадок чутливості самого рівняння, а чи не методу його розв'язання. Однак чутливі квадратні рівняння все ж таки можуть вирішуватися так точно, щоб задовольнити нашому критерію, якщо частину обчислень проводити з підвищеною точністю (в даному випадку має бути t = 16 або ще більше).

У певному сенсі коріння, обчислені у разі 5 (х1 = х2 = 2) є «хорошими» рішеннями цього рівняння. Зауважимо, що вони будуть точним корінням рівняння

*x2-*4х +4 = 0.

Це те саме рівняння, що і у випадку 5, за винятком вільного члена, що відрізняється від одиниці останнього розряду. Таким чином, ми вирахували точне коріння рівняння, дуже «близького» до того, яке намагалися вирішити.

Цей останній спосіб обліку помилок округлення називається зворотним аналізом; він був великою мірою розроблений Дж. Вілкінсон. Говорячи загалом, такий підхід характеризується наступним питанням: наскільки малі зміни у вихідних даних задачі необхідні для того, щоб подати обчислені результати як точне вирішення обуреного завдання. У класичному підході до помилок округлення у прямому аналізі з'ясовується просто, наскільки помилковими є отримані відповіді як розв'язання задачі з її власними вихідними даними. Хоча обидва методи корисні, важливою особливістю погляду зворотного аналізу є те, що він дозволяє аналізувати помилки округлення великих матричних або поліноміальних завдань, оскільки допускає використання асоціативних операцій, що часто дуже важко у прямому аналізі помилок.

Незважаючи на елементарний характер квадратного рівняння, можна з великою часткою впевненості сказати, що (принаймні на момент написання цієї книги) у всьому світі було не більше п'яти машинних програм, що задовольняють наміченим вище критеріям. Створення такого алгоритму не є дуже глибоким завданням, але все ж таки воно вимагає ясного усвідомлення мети та уваги до деталей при досягненні цієї мети. Ось сорт завдань, де молодшекурсник математичного або обчислювального факультету може зробити істотний внесок у машинні бібліотеки програм.

# 2 Формування випадкових сигналів (перешкод)

Популярність стохастичних алгоритмів зростає. Багато хто з них базується на генерації великої кількості різних випадкових величин. Не завжди рівномірно розподілених. Тут зібрана інформація про швидкі та точні генератори випадкових величин з відомими розподілами. Завдання можуть бути різними, різними можуть бути критерії. Комусь важливий час генерації, комусь – точність, комусь – криптостійкість, комусь – швидкість збіжності. Ми виходимо з припущення, що є певний базовий генератор, що повертає псевдовипадкове ціле число, рівномірно розподілене від 0 до деякого RAND\_MAX

unsigned long long BasicRandGenerator()

{

unsigned long long randomVariable;

// some magic here

...

return randomVariable;

}

і що цей генератор є досить швидким. Мається на увазі, що дешевше згенерувати з десяток випадкових чисел, ніж порахувати логарифм або звести в ступінь одне з них. Це можуть бути стандартні генератори: std::rand(), rand в MATLAB, Java.util.Random і т.д. Але майте на увазі, що такі генератори рідко підходять для серйозної роботи. Найчастіше вони провалюють різні статистичні випробування. А також, пам'ятайте, що ви повністю залежите від них і краще використовувати власний генератор, щоб мати уявлення про його роботу.

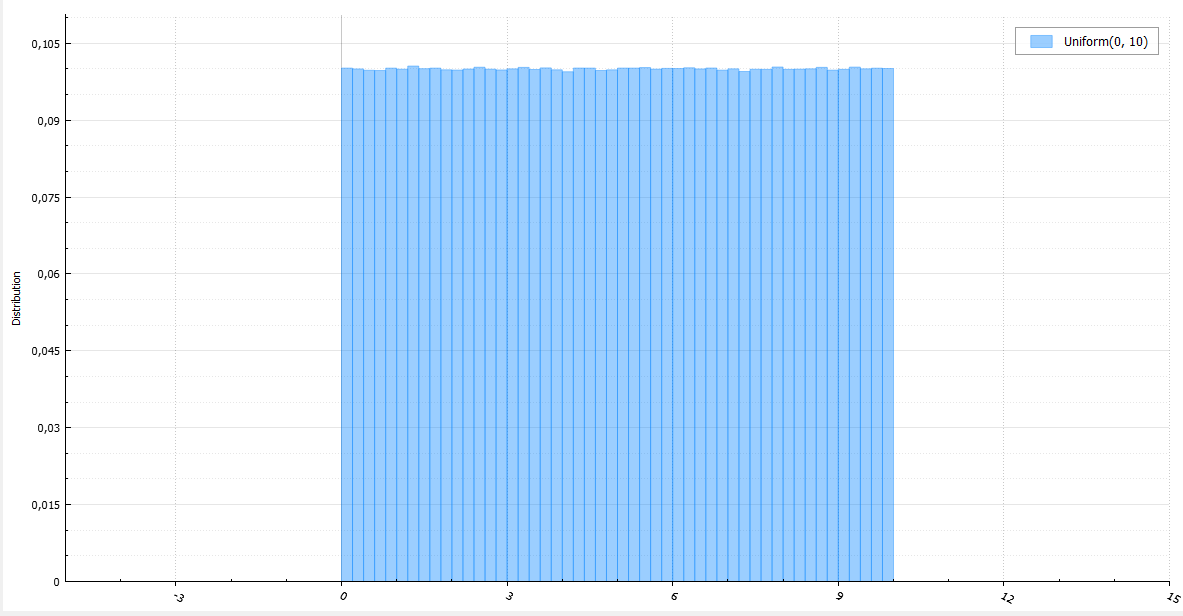
Далі йтиметься про алгоритми, суть яких має бути зрозумілою кожному, хто хоч іноді стикався з теорією ймовірностей. Зовсім необов'язково бути знайомим з теорією заходи, як правило, досить приблизно розуміти, що собою функція розподілу і функція щільності розподілу:

https://habrastorage.org/r/w1560/files/591/8a7/1dc/5918a71dcb1e4a6bb432c273bc65541d.png

Кожен алгоритм супроводжуватиме кодом, невеликою кількістю математики та гістограмою з десятка мільйонів згенерованих випадкових величин.

## 2.1 Формування чисел із рівномірним розподілом

https://habrastorage.org/r/w1560/files/abf/358/849/abf35884902446349f8fd3516788f894.png



Рівномірний розподіл може використовуватися при генерації майже будь-якої випадкової величини, благо є дуже простий та універсальний метод інверсії (inverse transform sampling): генеруємо випадкову величину U, рівномірно розподілену від 0 до 1, і повертаємо зворотну функцію розподілу (квантиль) з параметром U. Дійсно:

https://habrastorage.org/r/w1560/files/db6/c26/038/db6c260383fb4971b4b2455942989543.png

Проблема у цьому, що підрахунок зворотної функції розподілу то, можливо довгим, якщо взагалі аналітично можливий.

Генератор рівномірного розподілу на інтервалі [a, b] є найпростішим і довго на ньому зупинятися не має сенсу.

Варто відзначити, що при великій кількості випадкових величин, що генеруються, краще порахувати (b — a) / RAND\_MAX тільки один раз:

double Uniform(double a, double b)

{

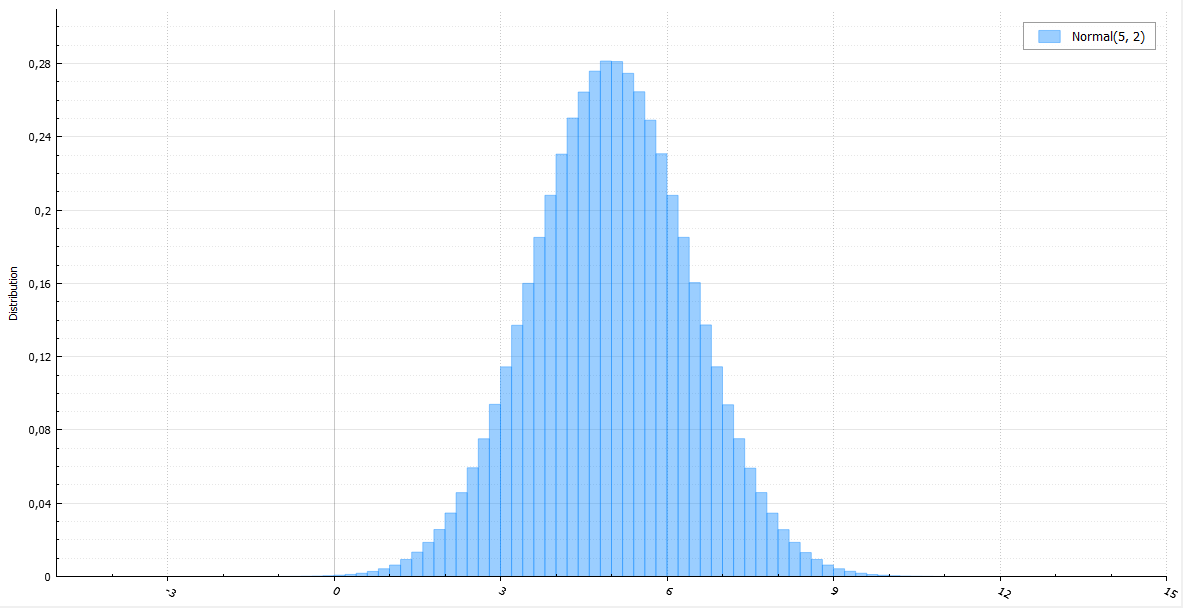
return a + BasicRandGenerator() \* (b - a) / RAND\_MAX;

}

Зрозуміло, безперервність це лише абстракція. У реальному світі й у разі саме під цим мається на увазі досить мінімальний крок дискретизації. Важливий момент. Якщо ви генеруєте випадкове число, рівномірно розподілене від 0 до 1, то 32 біт недостатньо, щоб покрити всі значення, які може приймати на цьому діапазоні double (хоча, для float цього більш ніж достатньо). Для кращої якості потрібно або генерувати 64-бітові цілі, або комбінувати два 32-бітові.

## 2.2 Формування чисел із нормальним розподілом

https://habrastorage.org/r/w1560/files/37d/dbf/53c/37ddbf53cb9647c7a05f56d103b48be8.png



Метод інверсії вимагатиме обчислення зворотної функції помилок. Якщо не використовувати спеціальні апроксимуючі функції, складно та неймовірно довго. Для нормальної величини існує метод Бокс-Мюллер. Він досить простий і поширений. Його явний недолік це обчислення трансцендентних функцій. Куди швидше працює красиво названий метод Ziggurat, придуманий Джорджем Марсалом, автором того ж полярного методу.

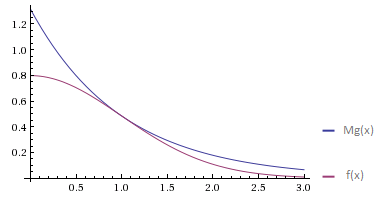
Полярний метод — приклад вибірки з відхиленням (acceptance-rejection sampling). Буквально, ви генеруєте величину і приймаєте її, якщо вона підходить, інакше відхиляєте і генеруєте ще раз. Основний приклад: потрібно згенерувати випадкову величину із щільністю f(x), проте це дуже складно зробити простим методом інверсії. Зате ви можете згенерувати випадкову величину з щільністю g(x), не дуже відрізняється від f(x). У такому разі ви берете найменшу константу M, таку, що M > 1 і майже всюди f(x) < Mg(x), і виконуєте наступний алгоритм:

— Генеруємо випадкову величину X із розподілу g(x) та випадкову величину U, рівномірно розподілену від 0 до 1.

- Якщо U<f(X) / Mg(X) - повертаємо X, інакше - повторюємо наново.

Імовірність прийняти випадкову величину:

https://habrastorage.org/r/w1560/files/f36/9f9/1a2/f369f91a23a74cfba2c052d76f8bffb8.png

Чим менше ви виберете M – тим більша ця ймовірність – тим швидше буде працювати генератор.

Наприклад згенеруємо модуль стандартної нормальної величини. З огляду на симетрії нормального розподілу отриману величину можна помножити на випадкову величину, що приймає значення +1 і -1 з рівними ймовірностями, і таким чином отримати стандартну нормально розподілену величину X. Будь-яка нормальна величина виходить зі стандартної множенням на sigma і зрушенням на mu. Функція щільності розподілу | X |:

https://habrastorage.org/r/w1560/files/7fc/abf/6f6/7fcabf6f6332427d9bb768e5d523072b.png

Спробуємо наблизити її функцією щільності стандартного експоненційного розподілу. Ми трохи забігаємо вперед, бо про експонентний розподіл ще не говорили. Воно генерується просто горезвісним методом інверсії - беремо рівномірно розподілену на [0, 1] величину U і повертаємо -ln (U).

https://habrastorage.org/r/w1560/files/335/83c/269/33583c269a314210b86f1574f34c8f5b.png

Мінімальне значення М, що відповідає умові f(x) < Mg(x), досягається при x = 1:

https://habrastorage.org/r/w1560/files/f29/c1a/1f9/f29c1a1f9b9c442e91cd8503038c5dee.png

Тоді:

https://habrastorage.org/r/w1560/files/16d/9a6/76c/16d9a676c3d44c2c90b476de53e499a7.png

І алгоритм для генерації нормальної випадкової величини буде наступним:

1) Генеруємо U - рівномірно розподілену на [0, 1] випадкову величину.

2) Генеруємо експоненційно розподілену випадкову величину E (про алгоритм, швидший, ніж підрахунок -ln(U), йтиметься далі).

3) Якщо U < exp(-(E - 1) 2) - то | X | = E, інакше повертаємось на перший крок.

4) Генеруємо нову U. Якщо U < 0.5, то повертаємо - | X |, або | X | в іншому випадку.

Неважко помітити, що:

https://habrastorage.org/r/w1560/files/e94/058/7e4/e940587e48164083b303ab73b4fbc8e5.png

А це означає, що замість U можна згенерувати ще одну експонентно розподілену величину і не вважати показову функцію. Новий алгоритм виглядатиме так:

1) Генеруємо стандартні експоненційно розподілені випадкові величини E1 і E2.

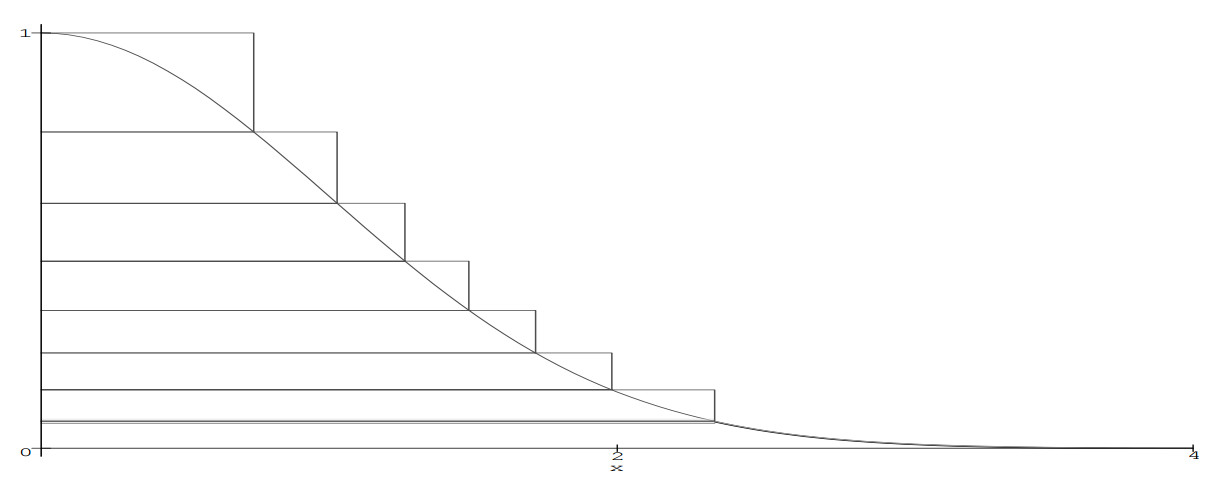
2) Якщо E2 > (E1 - 1) 2/2, то приймаємо | X | = E1, інакше повертаємось назад.

3) Генеруємо U. Якщо U < 0.5, то повертаємо - | X |, або | X | в іншому випадку.

Ще один момент: за умови прийняття величини E1 різниця E2 - (E1 - 1) 2/2 буде також розподілена експоненційно і незалежно від E1. Тому її можна запам'ятати та використати наступного разу замість E1.

Загальна проблема вибірки з відхиленням полягає у підборі такої випадкової величини з щільністю розподілу g(x), щоб відхилень було якнайменше. Для вирішення цієї проблеми існує багато розширень. Сам же метод є основою для багатьох наступних алгоритмів, включаючи Ziggurat. Суть останнього все та ж: намагаємося покрити функцію щільності нормального розподілу схожою та простішою функцією та повертаємо величини, що потрапили під криву. Функція своєрідна і нагадує багатоступінчасту споруду, звідки, власне, і така назва алгоритму.

### 2.2.1 Метод Ziggurat

Зіккурат споруджується в такий спосіб. У підніжжя функції f(x) вибираються точки x1 та y0 = f(x1). Площа під прямокутником від (0,0) до (x1, y0) + площа під хвостом функції f(x > x1) = А. Так ми збудували базовий шар. Поверх нього ставиться ще один прямокутник, такий, що його ширина x1, а висота y1 = A/y0 і таким чином його площа дорівнюватиме A. Цейпрямокутник вже включає точки, які лежать вище функції f(x), наприклад (x1, y1). Функцію f(x) другий прямокутник перетинає в точці (x2, y1) — це буде координата нижньої правої точки третього прямокутника, який накладається так само як і другий, щоб його площа дорівнювала А. Так триває доти, доки ми не Збудуємо Зіккурат до вершини функції. Площа кожного ступеня дорівнюватиме А. Подальший алгоритм (без обробки попадання в базовий шар):

1) Випадково та рівномірно вибирається прямокутник i та генерується рівномірно розподілена величина X від 0 до xi

2) Якщо X < xi+1, ми точно потрапили під криву — повертаємо X.

3) Генеруємо нову рівномірно розподілену величину Y від yi до yi+1. Якщо точка з координатами (X, Y) лежить під кривою, тобто Y<f(X) - повертаємо X. Інакше повертаємось на перший крок.

Що робити, якщо потрапили в базовий шар? З нього також можна зробити прямокутник із площею А, ввівши уявну точку x0 = A/y0. Тоді площа прямокутника від (0,0) до (x0, y0) також дорівнює А. Різниця в базовому шарі в тому, що якщо ми не потрапили під криву в новому уявному прямокутнику, то ми ще не повинні відхиляти x — ми лише потрапили у хвіст. Для хвоста Джордж Марсалья пропонує скористатися тим самим методом, що ми розглядали до цього - вибіркою з відхиленням. Тільки з деякими змінами:

1) Генеруємо стандартні експоненційно розподілені випадкові величини E1 із щільністю x1 та E2 з одиничною щільністю.

2) Якщо E2 > E12/2, то приймаємо | X | = E1 + x1, інакше повертаємось назад.

3) Генеруємо U. Якщо U < 0.5, то повертаємо - | X |, або | X | в іншому випадку.

Тоді повністю алгоритм виглядатиме так:

1) Випадково та рівномірно вибирається прямокутник i та генерується рівномірно розподілена величина X від 0 до xi.

2) Якщо X<xi+1, то ми точно потрапили під криву - повертаємо X.

3) Якщо потрапили до базового шару, тобто i = 0, то запускаємо алгоритм для хвоста і повертаємо його результат.

4) Генеруємо нову рівномірно розподілену величину Y від yI до yi + 1. Якщо точка з координатами (X, Y) лежить під кривою, тобто Y<f(X) - повертаємо X. Інакше повертаємось на перший крок.

Питання, як знайти константи А і x1 залежно від числа прямокутників? Можна їх визначити звичайним чисельним способом. Беремо початкове припущення x1, будуємо базовий шар, обчислюємо А і будуємо зіккурат, поки не побудували потрібну кількість прямокутників. Якщо ми виявилися вищими за функцію — значить ми взяли надто маленьке значення x1 і надто велике значення А — намагаємось побудувати з новими значеннями заново.

Сам алгоритм можна прискорити різними хитрощами роботи з 32-бітовими цілими, забираючи непотрібні перемноження. Для отримання більш детальної інформації можна звернутися до статті George Marsaglia та Wai Wan Tsang “Зигурний метод для генерації Random Variables”. Тут ми представляємо не до кінця оптимізований, зате зрозумілий код (з урахуванням, що ми вміємо генерувати експоненційний розподіл).

static double stairWidth[257], stairHeight[256];

const double x1 = 3.6541528853610088;

const double A = 4.92867323399e-3; /// area under rectangle

void setupNormalTables()

{

// Coordinates of the implicit rectangle in base layer

stairHeight[0] = exp(-.5 \* x1 \* x1);

stairWidth[0] = A/stairHeight[0];

// implicit value for the top layer

stairWidth[256] = 0;

for (unsigned i = 1; i <= 255; ++i)

{

// such x\_i that f(x\_i) = y\_{i-1}

stairWidth[i] = sqrt(-2 \* log(stairHeight[i - 1]));

stairHeight[i] = stairHeight[i - 1] + A/stairWidth[i];

}

}

double NormalZiggurat()

{

int iter = 0;

do {

unsigned long long B = BasicRandGenerator();

int stairId = B & 255;

double x = Uniform(0, stairWidth[stairId]); // get horizontal coordinate

if (x < stairWidth[stairId + 1])

return ((signed) B > 0) ? x: -x;

if (stairId == 0) // handle the base layer

{

static double z=-1;

double y;

if (z > 0) // we don't have to generate another exponential variable as we already have one

{

x = Exponential (x1);

z -= 0.5\*x\*x;

}

if (z <= 0) // if previous generation wasn't successful

{

do

{

x = Exponential (x1);

y = Exponential(1);

z = y - 0.5 \* x \* x; // we storage this value as after aceptance it becomes exponentially distributed

} while (z<=0);

}

x + = x1;

return ((signed) B > 0) ? x: -x;

}

// handle the wedges of other stairs

if (Uniform(stairHeight[stairId - 1], stairHeight[stairId]) < exp(-.5 \* x \* x))

return ((signed) B > 0) ? x: -x;

} while (++iter <= 1e9); /// one billion should be enough

return NAN; /// fail due to some error

}

double Normal(double mu, double sigma)

{

return mu + NormalZiggurat() \* sigma;

}

### 2.2.2 Метод підсумовування

Формування випадкових величин, розподілених за нормальним законом, можна виконати на підставі центральної граничної теореми теорії ймовірностей:

розподіл ймовірностей величини x,рівної сумі n незалежних випадкових величин βi

з довільним законом розподілу і дисперсіями, що мало розрізняються (тобто дають приблизно однаковий внесок у загальну суму) при n → ∞ асимптотично прагне до нормального закону.

Якщо величини βi рівномірно розподілені на інтервалі [0…1], то:

- математичне очікування випадкової величини x дорівнює Mx=n/2;

- Дисперсія дорівнює Dx = n/12.

Насправді вже за n>=8 розподіл формованої таким чином випадкової величини x досить близько до нормального.

Якщо при формуванні випадкової величини виконати нормування:



то x буде розподілено за нормованим нормальним законом з Mx=0 і Dx=σ=1.

## 2.3 Формування чисел з експонентним розподілом

https://habrastorage.org/r/w1560/files/5b8/8e6/80d/5b88e680db99481eab2ddd004b75ee6b.png

Вже говорилося, що з експоненціального розподілу можна взяти логарифм від рівномірно розподіленої величини і можна зробити генерацію швидше. Оскільки будь-яка експоненційна величина виходить зі стандартної поділом на щільність, генерацію можна зробити горезвісним Ziggurat. У разі попадання в хвіст можна запустити алгоритм за новою та додати до отриманої величини x1:

Ще кілька фактів: якщо використовувати таблицю з 255 прямокутниками, то ймовірність прийняття з першого разу для експоненційного розподілу — 0.989, для нормального — 0.993. У MATLAB із 5 версії для нормального розподілу використовується Ziggurat (раніше використовувався полярний метод).



static double stairWidth[257], stairHeight[256];

const double x1 = 7.69711747013104972;

const double A = 3.9496598225815571993e-3; /// area under rectangle

void setupExpTables()

{

// Coordinates of the implicit rectangle in base layer

stairHeight[0] = exp(-x1);

stairWidth[0] = A/stairHeight[0];

// implicit value for the top layer

stairWidth[256] = 0;

for (unsigned i = 1; i <= 255; ++i)

{

// such x\_i that f(x\_i) = y\_{i-1}

stairWidth[i] = -log(stairHeight[i - 1]);

stairHeight[i] = stairHeight[i - 1] + A/stairWidth[i];

}

}

double ExpZiggurat()

{

int iter = 0;

do {

int stairId = BasicRandGenerator() & 255;

double x = Uniform(0, stairWidth[stairId]); // get horizontal coordinate

if (x < stairWidth[stairId + 1]) /// if we are under the upper stair - accept

return x;

if (stairId == 0) // if we catch the tail

return x1 + ExpZiggurat();

if (Uniform(stairHeight[stairId - 1], stairHeight[stairId]) < exp(-x)) // if we are under the curve - accept

return x;

// rejection - go back

} while (++iter <= 1e9); // one billion should be enough to be sure there is a bug

return NAN; // fail due to some error

}

double Exponential(double rate)

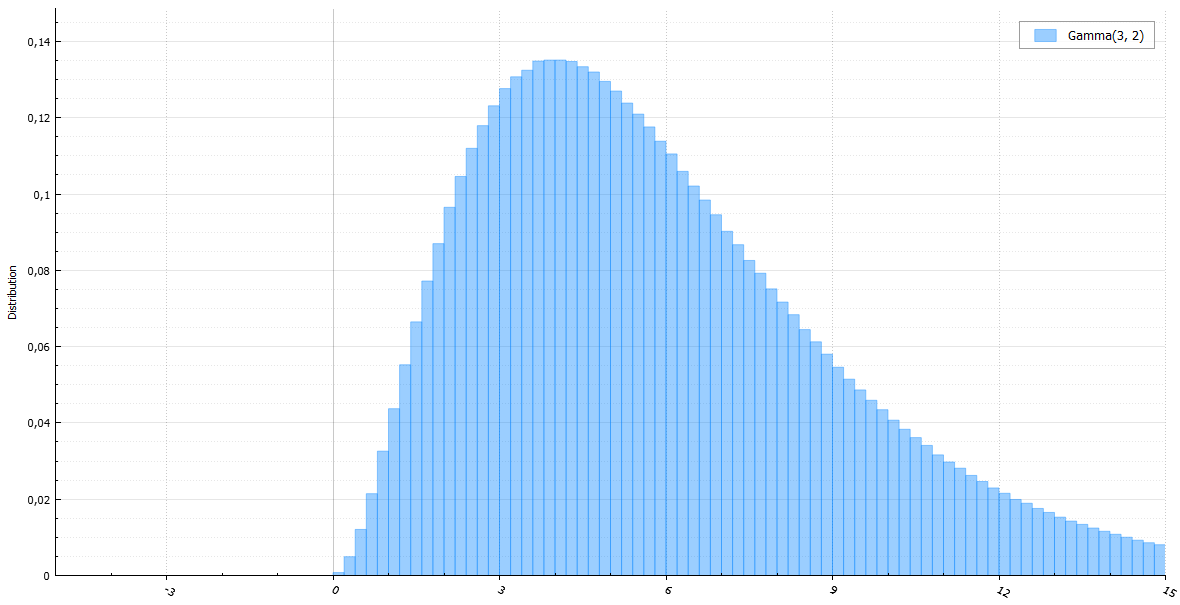
{

return ExpZiggurat() / rate;

}

## 2.4 Формування чисел із гамма-розподілом

https://habrastorage.org/r/w1560/files/42a/771/a70/42a771a70a5e483ab100b659e7a9b769.png



Алгоритми для генерації тут складніше. Для генерації стандартної величини (theta = 1) використовуються чотири алгоритми, кожен залежно від k.

Якщо k або 2k — ціле і k<5 — GA алгоритм.

Якщо k<1 — GS алгоритм.

Якщо 1 < k < 3 – GF алгоритм.

Якщо k > 3 — алгоритм GO.

Нестандартна випадкова величина з гамма-розподілом, що виходить зі стандартної множенням на theta.

### 2.4.1 Алгоритм GA

Якщо скласти дві випадкові величини з гамма-розподілом з параметрами k1 та k2, то вийде випадкова величина з гамма-розподілом та з параметром k1+k2. Ще одна властивість - якщо theta = k = 1, то легко перевірити, що розподіл буде експонентним. Тому, якщо k ціле — можна просто підсумувати k випадкових величин зі стандартним експоненційним розподілом.

double GA1(int k)

{

double x = 0;

for (int i = 0; i < k; ++i)

x += Exponential(1);

return x;

}

Якщо k не ціле, але 2k — ціле, можна замість однієї з експоненційних випадкових величин у сумі використовувати половину квадрата нормальної величини. Чому так можливо, стане зрозуміло пізніше.

double GA2(double k)

{

double x = Normal (0, 1);

x \* = 0.5 \* x;

for (int i = 1; i < k; ++i)

x += Exponential(1);

return x;

}

### 2.4.2 Алгоритм GS

1) Генеруємо стандартну експоненційно розподілену величину Е та величину U, рівномірно розподілену від 0 до 1 + k/e. Якщо U <= 1, переходимо до кроку 2. Інакше, до кроку 3.

2) Задаємо x = U1/k. Якщо x<=E, то повертаємо x, інакше повертаємось назад на крок 1.

3) Задаємо x = -ln ((1 - U) / k + 1 / e). Якщо (1 - k) \* ln (x) <= E, то повертаємо x, інакше повертаємось на крок 1.

double GS(double k)

{

// Assume that k < 1

double x = 0;

int iter = 0;

do {

// M\_E is base of natural logarithm

double U = Uniform (0, 1 + k / M\_E)

double W = Exponential(1)

if (U<=1)

{

x = pow(U, 1.0/k);

if (x <= W)

return x;

}

else

{

x = -log((1 - U)/k + 1.0/M\_E);

if ((1 - k) \* log(x) <= W)

return x;

}

} while (++iter < 1e9); // excessive maximum number of rejections

return NAN; // shouldn't end up here

}

### 2.4.3 Алгоритм GF

Наскільки відомо, автор цього алгоритму, професор Університету Північної Кароліни Дж. С. Фішман не став опубліковувати своє досягнення. Сам алгоритм працює для k > 1, проте середній час його виконання зростає пропорційно sqrt(k), тому він ефективний лише для k < 3. Алгоритм:

1) Генеруємо дві незалежні випадкові величини E1 і E2 зі стандартним експоненційним розподілом.

2) Якщо E2 < (k - 1) \* (E1 - ln (E1) - 1), то повертаємось назад на крок 1.

3) Повертаємо x = k \* E1

double GF(double k)

{

// Assume that 1 < k < 3

double E1, E2;

do {

E1 = Exponential(1);

E2 = Exponential(1);

} while (E2 <(k - 1) \* (E1 - log(E1) - 1));

return k\*E1;

}

### 2.4.4 Алгоритм GO

Алгоритм Дітера та Аренса заснований на асимптотичній нормальності розподілу при збільшенні параметра, і тому швидший для великих k. Він працює для k > 2.533, проте менш ефективний як алгоритм Фішера для k < 3. Спочатку потрібно задати деякі константи (залежать лише від k).

void setupConstants(double k)

{

m = k – 1;

s\_2 = sqrt (8.0 \* k / 3) + k;

s = sqrt (s\_2);

d = M\_SQRT2 \* M\_SQRT3 \* s\_2;

b = d+m;

w = s\_2/(m – 1);

v = (s\_2 + s\_2) / (m \* sqrt (k));

c = b + log(s \* d / b) – m – m – 3.7203285;

}

double GO(double k)

{

// Assume that k > 3

double x = 0;

int iter = 0;

do {

double U = Uniform (0, 1);

if (U<=0.0095722652) {

double E1 = Exponential(1);

double E2 = Exponential(1);

x = b\*(1+E1/d);

if (m \* (x / b - log(x / m)) + c <= E2)

return x;

}

else {

double N;

do {

N = Normal (0, 1);

x = s \* N + m; // ~ Normal (m, s)

} while (x < 0 | | x > b);

U = Uniform (0, 1);

double S = 0.5 \* N \* N;

if (N > 0) {

if (U < 1 - w \* S)

return x;

}

else if (U < 1 + S \* (v \* N - w))

return x;

if (log(U) < m \* log(x / m) + m - x + S)

return x;

}

} while (++iter < 1e9);

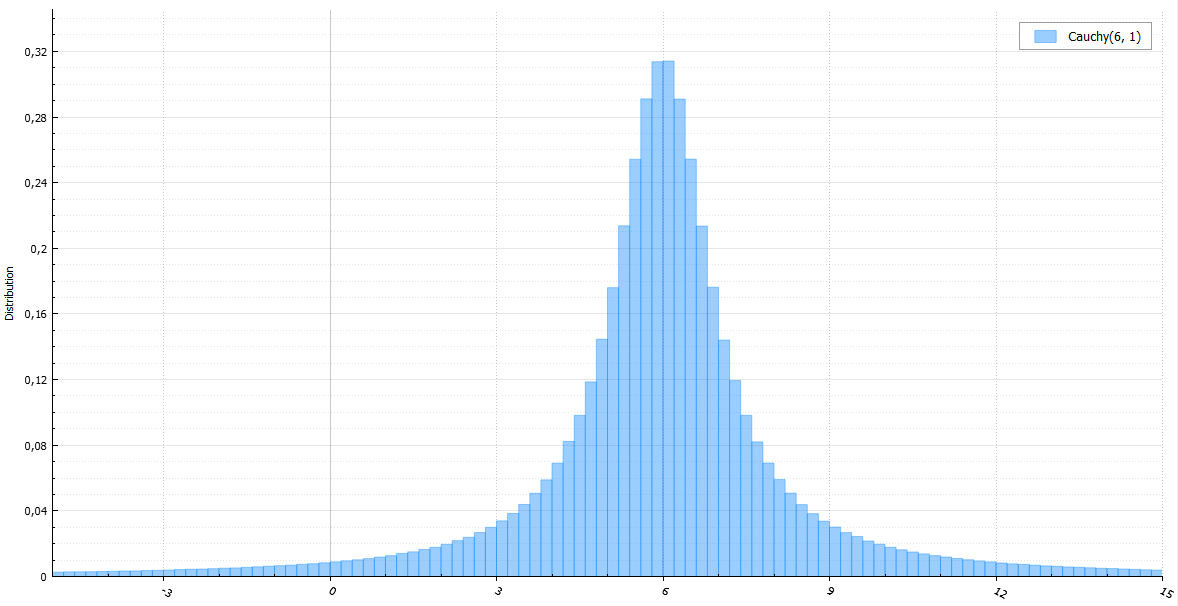
return NAN; // shouldn't end up here;

}

Як і у випадку попередніх розподілів, будь-яка величина з гамма-розподілом генерується зі стандартної множенням на параметр theta. Вміючи генерувати рівномірний, нормальний, експоненційний і гамма-розподіл, ви, швидше за все, з легкістю зможете отримати більшість інших розподілів, благо вони легко виражаються через вищезгадані.

## 2.5 Формування чисел із розподілом Коші

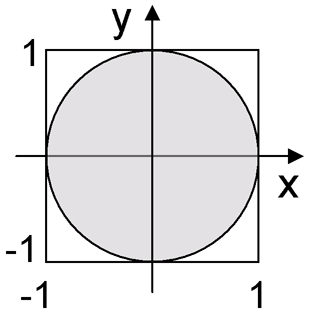
https://habrastorage.org/r/w1560/files/534/f41/144/534f41144b454f34a522b273a3e62c0a.png



Один із способів швидко отримати випадкову величину з розподілом Коші – це взяти дві нормальні випадкові величини та поділити один на одного. Чи можна швидше? Можна скористатися звичайним методом інверсії, але вимагатиме обчислення тангенса. Це ще повільніше.

https://habrastorage.org/r/w1560/files/2f9/72b/c3f/2f972bc3fdb8403f883d4a55949bb436.png

Чи ні?

Пам'ятаєте полярний метод Джорджа Марсальї? Він дозволяв у перетворенні Бокса-Мюллера швидко отримувати синус чи косинус рівномірно розподіленої випадкової величини. Так само можна отримати і тангенс. Генеруємо дві рівномірно розподілені випадкові координати x та y на квадраті [-1, 1]x[-1, 1]. Якщо ми потрапили в коло з центром (0, 0) і одиничним радіусом — повертаємо x/y, інакше — генеруємо x та y ще раз. Імовірність не потрапити в коло, скажімо, з третього разу вже менше ніж 0.01.

double Cauchy(double x0, double gamma) {

double x, y;

do {

x = Uniform(-1,1);

y = Uniform(-1,1);

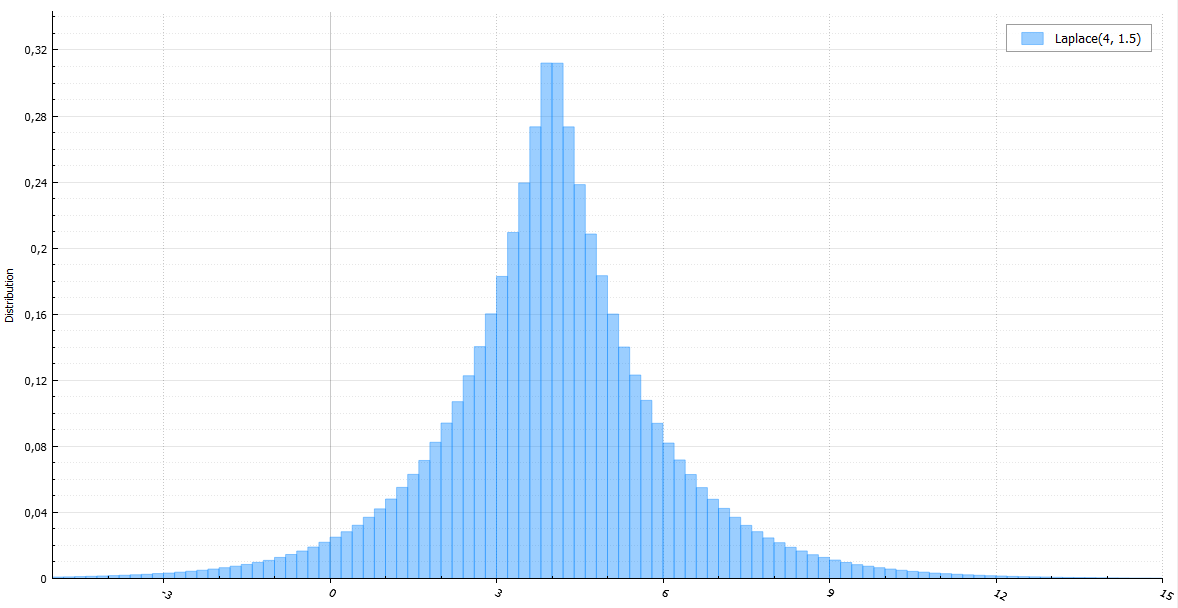
} while (x \* x + y \* y > 1.0 | | y = = 0.0);

return x0 + gamma \* x/y;

}

## 2.6 Формування чисел із розподілом Лапласа

https://habrastorage.org/r/w1560/files/492/d26/1ee/492d261eeac04942ac6ee989dc9e13e1.png



Розподіл Лапласа - це той самий експоненційний, тільки з випадковим знаком.

double Laplace(double mu, double b)

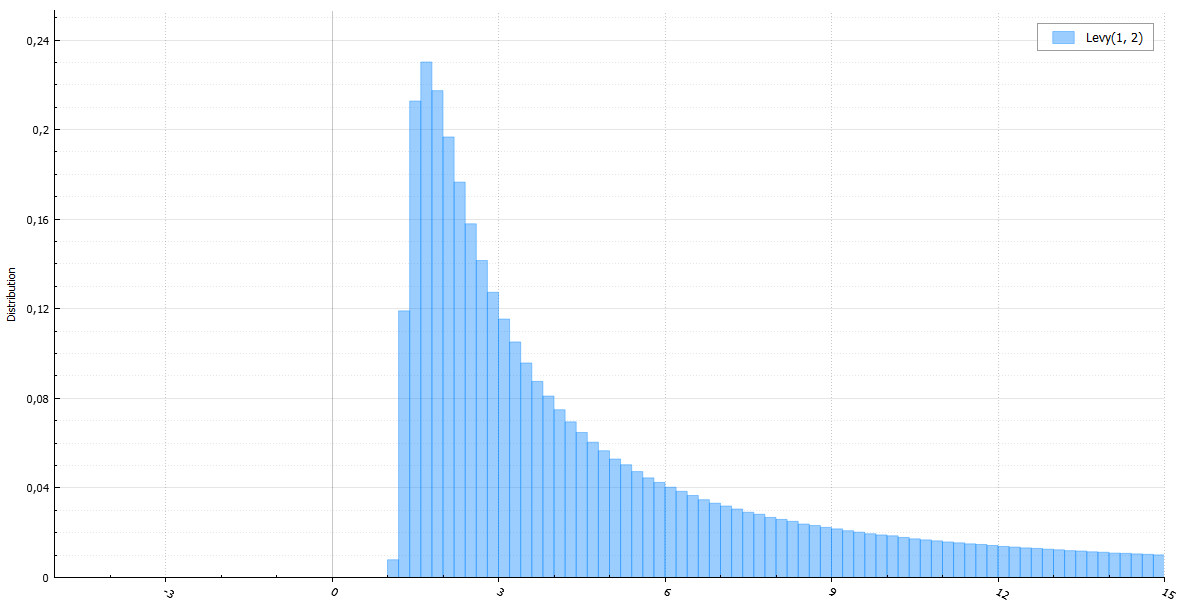
{

double E = Exponential(1.0/b);

return mu + (((signed)BasicRandGenerator() > 0) ? E : -E);

}

## 2.7 Формування чисел з розподілом Леві

https://habrastorage.org/r/w1560/files/7ce/b35/45f/7ceb3545f08f4706817be626042a39a2.png

Незважаючи на те, як жахливо виглядає функція щільності розподілу, сама випадкова величина генерується дуже просто і швидко:

double Levy(double mu, double c)

{

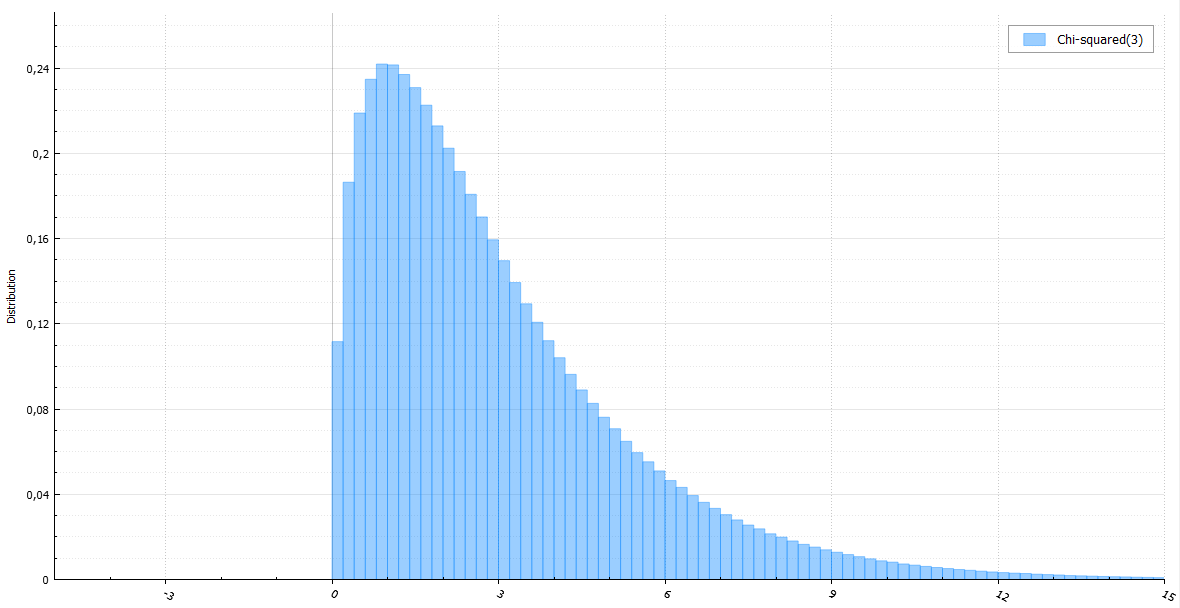
double N = Normal (0, 1);

return mu + c/(N\*N);

}

## 2.8 Формування чисел із розподілом хі-квадрат

https://habrastorage.org/r/w1560/files/5df/251/ae2/5df251ae25474d25ac2f7e40a4e6ac71.png



Як відомо, випадкова величина із розподілом хі-квадрат із параметром k — це сума квадратів k стандартних нормальних випадкових величин. Менш відомо, що при парному k це ж і подвоєна сума k/2 експоненційних випадкових величин (або розподіл Ерланга). З цього співвідношення відбувається вищезгаданий алгоритм GA2:

double ChiSquared(int k)

{

// ~ Gamma (k / 2, 2)

if (k >= 10) // too big parameter

return GO(0.5\*k);

double x = ((k & 1) ? GA2(0.5 \* k) : GA1(k >> 1));

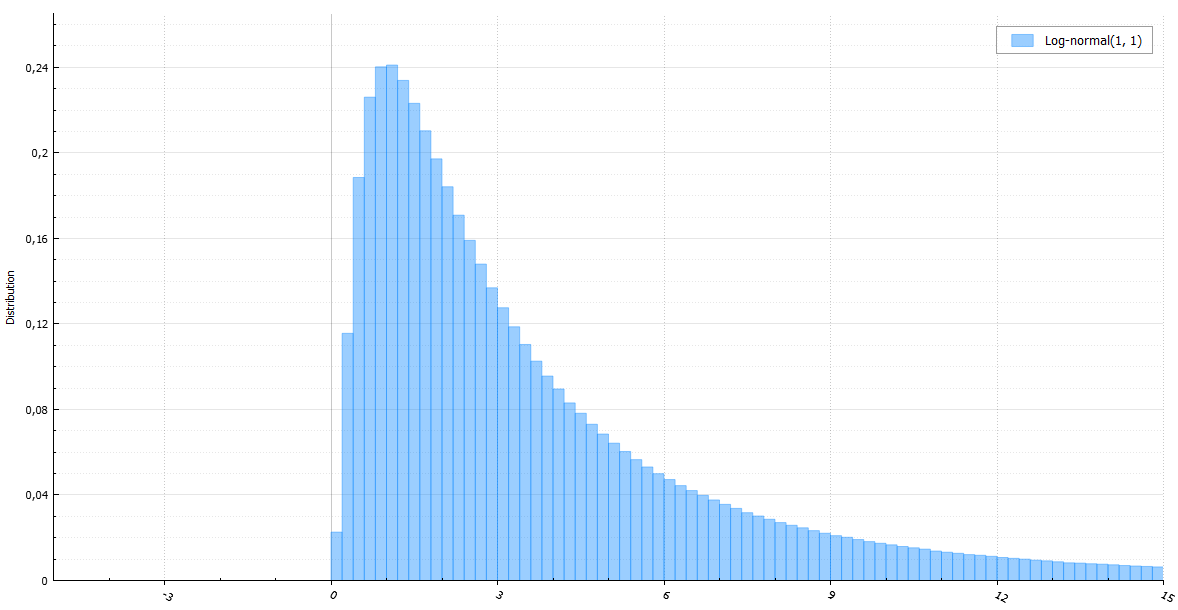
return x + x;

}

З цієї властивості слід примітний факт: якщо вибирати точку на площині з випадковими та нормальними координатами (x, y), то відстань від (0,0) до цієї точки буде розподілено експоненційно.

## 2.9 Формування чисел із логнормальним розподілом

https://habrastorage.org/r/w1560/files/e8c/31f/d40/e8c31fd4022d490dbe0e7b5f8f687298.png



Люди, які працюють з математичними моделями на фінансових ринках, про логнормальний розподіл знають не з чуток. Для логнормального, лог-кошу, лог-логістичного розподілів є один простий спосіб - взяти експоненту.

double LogNormal(double mu, double sigma)

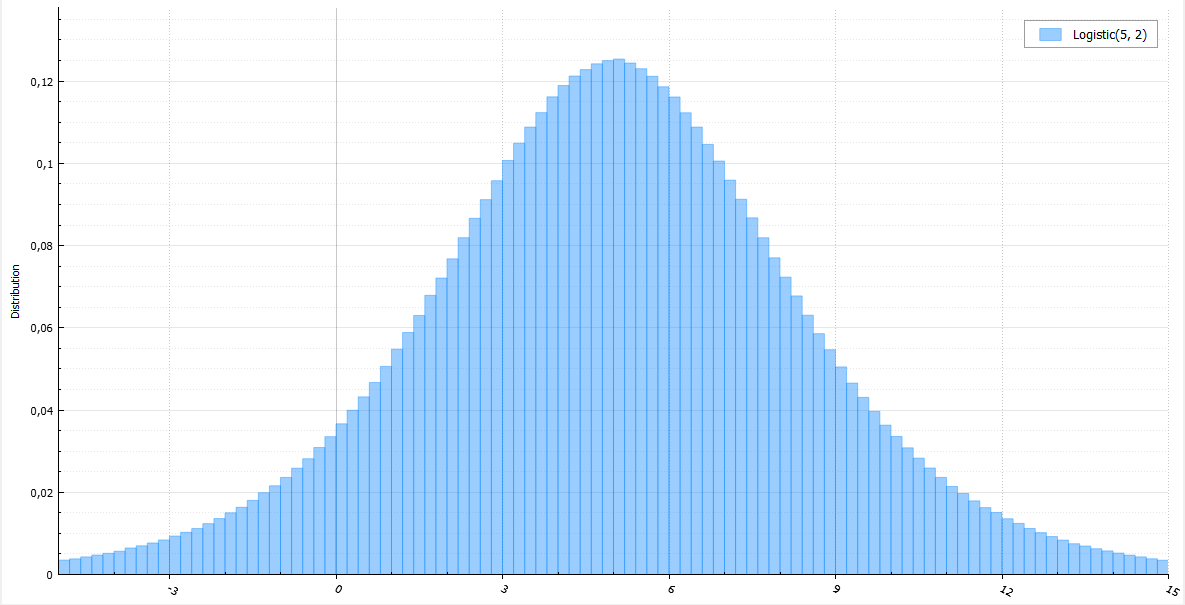
{

return exp(Normal(mu, sigma));

}

## 2.10 Формування чисел із логістичним розподілом

https://habrastorage.org/r/w1560/files/588/c68/3ef/588c683ef54743968b26d22238a6366d.png



Логістичний розподіл дуже схожий на нормальний, але має більш важкий хвіст, генерується досить легко через стандартний рівномірний розподіл:

double Logistic(double mu, double s)

{

return mu + s \* log (1.0 / Uniform (0, 1) - 1);

}

Насамкінець коротко про не завжди швидкі, але дуже прості алгоритми для інших популярних розподілів:

double Erlang(int k, double l)

{

return GA1(k)/l;

}

double Weibull(double l, double k)

{

return l \* pow(Exponential(1), 1.0/k);

}

double Rayleigh(double sigma)

{

return sigma \* sqrt (Exponential (0.5));

}

double Pareto(double xm, double alpha)

{

return xm/pow(Uniform(0, 1), 1.0/alpha);

}

double StudentT(int v)

{

if (v == 1)

return Cauchy (0, 1);

return Normal (0, 1) / sqrt (ChiSquared (v) / v);

}

double FisherSnedecor(int d1, int d2)

{

double numerator = d2 \* ChiSquared (d1);

double denominator = d1 \* ChiSquared (d2);

return numerator/nominotor;

}

double Beta(double a, double b) {

double x = Gamma (a);

return x/(x + Gamma(b));

}

# 3 Математична модель надійності обладнання

На практиці надійність будь-якого елемента визначається двома складовими: ймовірністю раптових відмов та ймовірністю зносових відмов. Випадкові відмови описуються експоненціальним законом, у якому надійність чи ймовірність безвідмовної роботи визначається виразом

, (3.1)(3.2)

де- Інтенсивність раптових відмов.

Для експоненційного випадку середнє напрацювання на відмову визначається як

. (3.2)(3.3)

У загальному випадку

. (3.3)(3.4)

Також для будь-якого закону розподілу справедлива формула

, (3.4)(3.5)

де- Щільність ймовірності відмов.

У період нормальної роботи для експонентного закону інтенсивність відмов є постійною величиною.

Коли настає зношування, інтенсивність відмов починає зростати і до раптових відмов додаються зносові, які зазвичай підкоряються нормальному закону розподілу.

, (3.5)(3.6)

де М - Середнє значення довговічності елемента з урахуванням зносу.

Стандартне відхилення від середньої довговічності визначається як

, (3.6)(3.7)

де N - Число відмов за час t.

Спільна ймовірність безвідмовної роботи елемента з урахуванням раптових та зносових відмов у період від t = 0, коли елемент новий, до часу t визначається виразом

. (3.7)(3.8)

Формулу (3.7) можна використовувати тільки для випадку, коли новий елемент. Якщо елемент вже має якесь напрацювання t0, то спільна ймовірність безвідмовної роботи визначається так:

. (3.8)(3.9)

Хоча вираз (3.8) дозволяє визначити надійність елемента в будь-який момент часу, воно має низку недоліків. Так, наприклад, факт увімкнення/вимкнення різного обладнання призводить до його форсованого зносу. Крім того, згідно з практичними спостереженнями, інтенсивність відмов суттєво залежить від зміни рівня навантаження елемента щодо її номінального значення. Коли навантаження елемента перевищує номінальне, спостерігається досить швидке зростання інтенсивності відмов. З іншого боку, інтенсивність відмов зменшується, коли навантаження стає нижчим за номінальний рівень.

Також добре відомо, що роботу елементів істотно впливають параметри довкілля. Ступінь опірності елементів впливам із боку довкілля можна умовно назвати «міцністю». "Міцність" елемента означає не тільки його опір механічним навантаженням, вібраціям, ударам, тиску або прискорення; до категорії міцності належать також: опір тепловим навантаженням, електрична міцність, вологостійкість, корозійностійкість, стійкість до випромінювань тощо. Тому очевидно, що міцність елемента може бути виражена деякою числової величиною.

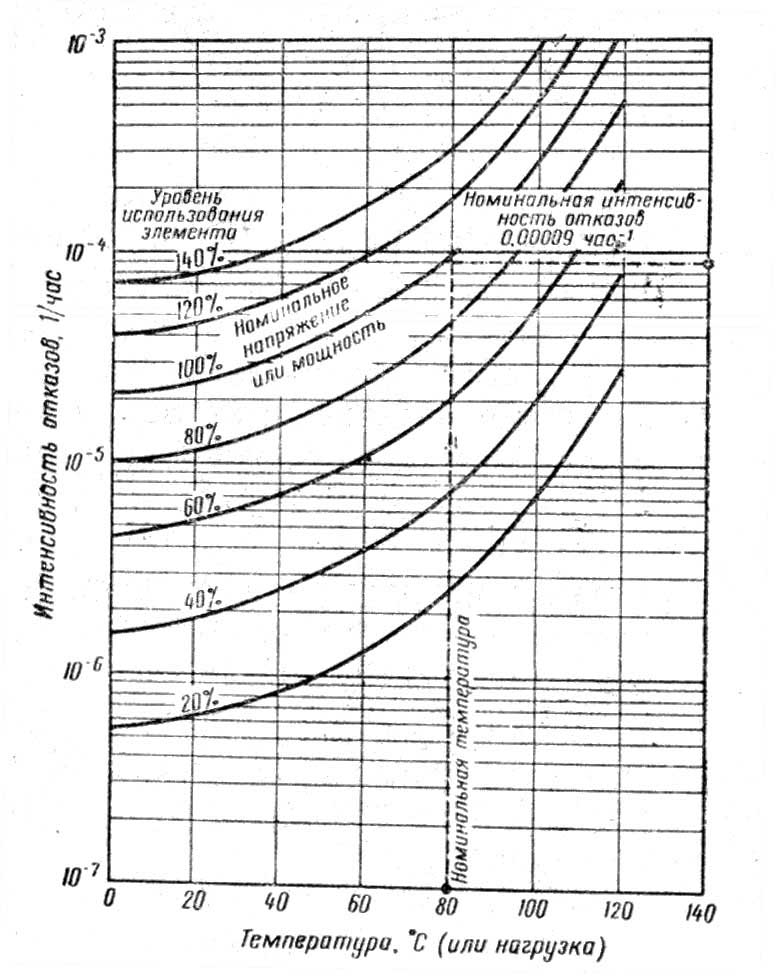


Рис. 3.1 Інтенсивність відмов у залежності від навколишніх умов

На практиці генеральну сукупність елементів піддають впливу робочого навантаження та визначають кількість елементів, які відмовилися протягом заданого періоду роботи. В результаті виходить оцінка інтенсивності відмов елемента заданого рівня навантаження. Потім можна по одному змінювати деякі параметри навколишнього середовища, наприклад збільшувати або зменшувати електричну напругу або температуру і повторювати весь експеримент в нових умовах. При цьому отримують інформацію про характер зміни інтенсивності відмов елементів, що розглядаються, від величини прикладеної напруги або температури навколишнього середовища. Таким чином, інтенсивність відмов замінює точну міру міцності елемента.

На рис. 3.1 наведено криві, що ілюструють загальний характер зміни інтенсивності відмов електричних та електронних та електромеханічних елементів.

Така ж тенденція спостерігається на практиці та інших видів елементів: механічних, теплотехнічних і т.д.

Оскільки описані фактори впливають насамперед на інтенсивність відмов, для моделювання надійності доцільно замість (3.8) використовувати вираз

, (3.9)(3.10)

де- Інтенсивність раптових відмов;

- Інтенсивність зносових відмов.

У період нормальної експлуатації інтенсивність раптових відмов постійна і визначається так:

. (3.10)(3.11)

Для визначення інтенсивності зносових відмов можна скористатися наступним підходом. З формули (3.4) у загальному випадку інтенсивність відмов визначається як. Розглянемо нагоду нормального розподілу. На рис. 3.2 зображені криві щільності нормального розподілу f(t), функції Pі(t) та інтенсивності відмов r(t). Розглянуто стандартизований варіант, при якому функція щільності відмов виражена в одиницях стандартного відхилення:

 (3.11)(3.12)

Значення φ(t) та P(t) можуть бути отримані безпосередньо з таблиць нормального розподілу.



Рис. 3.2 Інтенсивність відмов при нормальному розподілі:

*φ*(t) - стандартна функція щільності зносових відмов;

*Pі*(t) - можливість безвідмовної роботи при зносових відмових;

*r*(t) - стандартна крива інтенсивності відмов

Таким шляхом виходить стандартизована функція інтенсивності відмов:

 . (3.12)(3.13)

Тоді інтенсивність зносових відмов:

. (3.13)(3.14)

Розглянемо тепер таку ситуацію. В ході імітаційного моделювання надійності елемента на черговий ітерації поточний час роботи було збільшено на деякий крок Δt, а потім за формулою (3.9) було визначено значення інтенсивності поточних зносових відмов. Припустимо, що на ітерації, що розглядається, елемент був включений або вимкнений, що повинно було привести до його підвищеного зносу. Тоді для моделювання підвищеного зносу достатньо додати до поточного часу елемента певну величину Δtі, що відповідає величині зменшення ресурсу елемента при його включенні/вимкненні. Результат аналогічного моделювання подано на рис. 3.3. Кривавідповідає теоретичній зміні надійності для раптових відмов. Кривавизначає зміну надійності при зносі. Крива 1 показує, як змінювалася б надійність при спільній дії раптових та зносових відмов при номінальному режимі роботи. Крива 2 відповідає реальним змінам надійності елемента з урахуванням форсованого зносу.



Рис. 3.3 Надійність елемента:

1 - при нормальній експлуатації;

2 - при підвищеному зносі

У вираз (3.13) можна додати поправний коефіцієнт на поточну робочу потужністьта поправочний коефіцієнт на параметри навколишнього середовища:

, (3.14)(3.15)

де- Наведена робоча потужність;

- Наведений параметр навколишнього середовища (температура, частота, вологість тощо).

;,

де- поточна робоча потужність;

- Номінальна робоча потужність;

- поточне значення параметра навколишнього середовища;

- Номінальне значення параметра навколишнього середовища.

Для використання електричних і електронних елементів представленого на рис. 3.1

 (3.15)(3.16)

, (3.16)(3.17)

де- Наведена температура навколишнього середовища.



Рис. 3.4 Зміна кривої надійності елемента під час роботи на різних потужностях

На рис. 3.4 представлений результат моделювання кривих надійності елемента при його роботі різних рівнях потужності.

При визначенні надійності всієї системи в цілому необхідно пам'ятати, що при послідовному з'єднанні компонентів n надійність визначається як

, (3.17)(3.18)

а при паралельному

. (3.18)(3.19)

Допустимо вирішується завдання оптимізації за критерієм надійності для системи що складається з 3х технічних засобів, що паралельно підключаються. У будь-який момент часу може реалізовуватись будь-яка з можливих комбінацій включення обладнання. У цьому випадку необхідно вирішити задачу мінімізації ймовірності відмови обладнання

.

Враховуючи, що система складається з трьох об'єктів, що з'єднуються паралельно, то можливість відмови системи для поточної конфігурації обладнання має вигляд

, (3.21)

де- можливість безвідмовної роботи і-ї одиниці обладнання для поточної конфігурації системи в кінці заданого інтервалу часу*, а*- загальний час роботи цієї одиниці обладнання на інтервалі часу.обчислюється за такою формулою (3.9). При цьому враховується, що кожен елемент системи має деяке напрацюваннятому.

# 4 Математична модель ефективності теплотехнічного обладнання

Ефективність обладнанняможе бути обчислена за енергетичним ККД з наступною модифікацією.

Використання в технічній системі різних фізичних принципів одержання теплової енергії не дозволяє застосувати для оцінкиНормальний коефіцієнт корисного впливу. Перевагою ККД є його значення, що належить до інтервалу (0;1). Для приведення до рівності якості різних енергоносіїв (газ, електроенергія), що розглядаються, можна використовувати для оцінки ефективності обладнанняенергетичний ККДз урахуванням поправки Карно для приведення енергоносіїв, що надходять до системи та виходять із системи, до однієї якості енергетичної продукції.

У випадку з газовим котлом ККД із поправкою визначається як

,

де- Енергетичний ККД котла;

- Температура навколишнього середовища, К;

- Температура теплоносія на виході з котла, К.

Якщо теплонасосна установка (ТНУ) працює у режимі теплопостачання, товизначається так:

,

де— кількість теплоти, що виділяється на конденсаторі ТНУ на інтервалі часу, що розглядається., Дж;

- Температура фреону в конденсаторі, К;

- Потужність компресора ТНУ, Вт.

Якщо ТНУ працює у режимі холодопостачання, товизначається таким чином:

,

де- Питома кількість енергії, що виділяється у випарнику, Дж.

Ефективність системи теплопостачання визначається з урахуванням вхіднихта вихіднихпотоків енергії:.

Допустимо технічна система теплопостачання складається з двох газових котлів і теплонасосної установки, які можуть у будь-який момент часу підключатися паралельно в будь-якій конфігурації. Допустимо також, що глобально вирішується завдання мінімізації енергетичних втрат. У цьому випадку складова ефективності системи теплопостачання для поточної конфігурації є величиною втрат енергії і має вигляд

,

де- Теплова енергія, що надходить від i-го котла для теплопостачання, Дж;- теплова енергія, що надходить від ТНУ для теплопостачання, Дж;- час роботи i-го котла на інтервалі, с;- Витрата газу при роботі i-го котла за час, Кг/с;- Теплотворна здатність газу, Дж/кг;.— час роботи ТНУ на інтервалі, с;- Потужність ТНУ, Вт.

# 5 Методи однокритеріальної оптимізації

## 5.1 Загальні відомості про градієнтні методи

Перш ніж описувати метод, слід спочатку описати завдання, а саме: «Дані безліч K і функція f : K → R, потрібно знайти точку, таку щодля всіх, Що зазвичай записується наприклад ось так:



Теоретично зазвичай передбачається, що f - диференційована і опукла функція, а K - опукла безліч (а ще краще, якщо взагалі), це дозволяє дати якісь гарантії успішності застосування градієнтного спуску. Насправді градієнтний спуск успішно застосовується навіть коли завдання немає жодного з перелічених вище властивостей.

Допустимо поки що нам потрібно просто знайти мінімум одномірної функції



Ще в 17 столітті П'єром Ферма був придуманий критерій, який дозволяв вирішувати прості завдання оптимізації, а саме, якщо x\* — точка мінімуму f\* , то



де- Похідна f. Цей критерій базується на лінійному наближенні

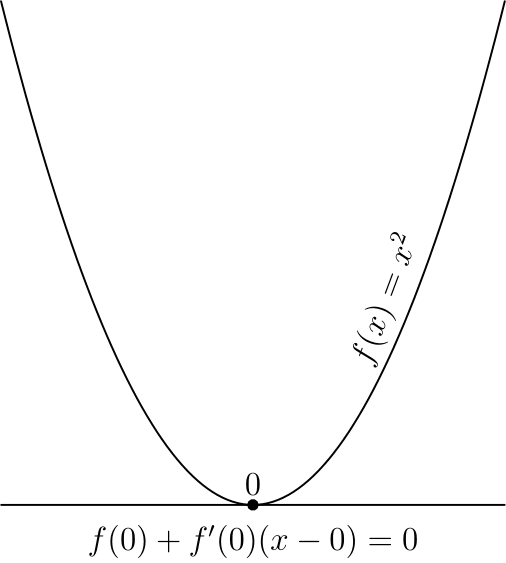
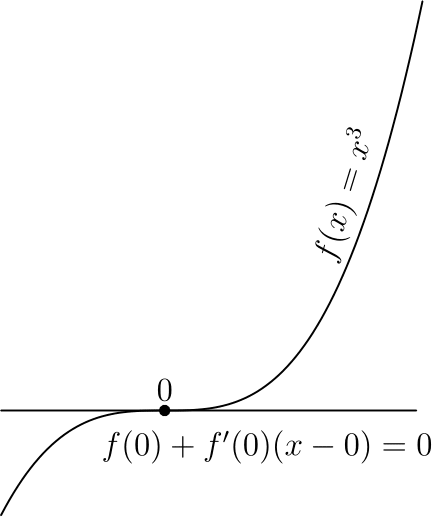


Чим ближче до x\*, тим точніше це наближення. У правій частині - вираз, який приможе бути як більшетак і менше – це основна суть критерію. У багатовимірному випадку аналогічно з лінійного наближення(тут і далі— стандартний скалярний твір, форма запису обумовлена ​​тим, що скалярний твір — це те саме, що матричний твір вектор-рядки на вектор-стовпець) виходить критерій:



Величина— градієнт функції f у точці x\*. Також рівність градієнта нуля означає рівність всіх приватних похідних нулю, тому в багатовимірному випадку можна отримати цей критерій просто послідовно застосувавши одновимірний критерій щодо кожної змінної окремо.

Варто зазначити, що зазначені умови є необхідними, але не достатніми, найпростіший приклад — 0 дляі

Цей критерій є достатнім у разі опуклої функції, багато в чому через це опуклі функції вдалося отримати так багато результатів.

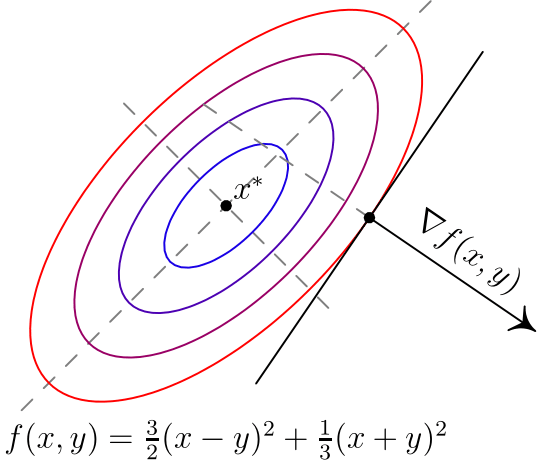
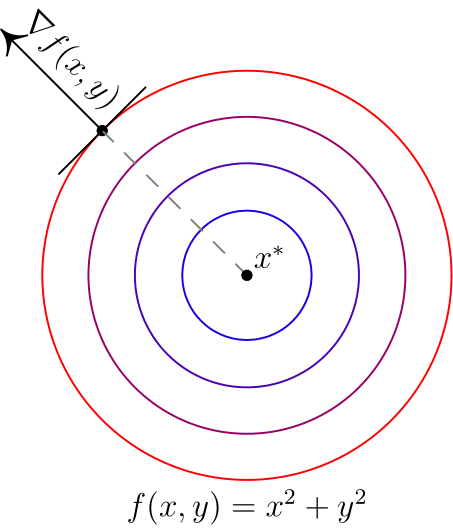
Отже, якщо функція диференційована (у неї існують похідні по всіх змінних), то в точці мінімуму градієнт повинен дорівнювати нулю. А ось чи несе градієнт якусь корисну інформацію у разі, коли він відмінний від нуля?

Спробуємо поки вирішити просте завдання: дана точка x, знайти точку таку, що. Давайте візьмемо крапку поряд з x, знову ж таки використовуючи лінійне наближення. Якщо взяти, то ми отримаємо



Аналогічно, якщотобуде більше(тут і далі). Знову ж таки, оскільки ми використали наближення, то ці міркування будуть вірні лише для малих. Підсумовуючи вищесказане, якщо, то градієнт показує напрямок найбільшого локального збільшення функції.

Нижче наведено два приклади для двовимірних функцій. Такі картинки можна часто побачити в демонстраціях градієнтного спуску. Кольорові лінії - так звані лінії рівня, це безліч точок, для яких функція набуває фіксованих значень, в даному випадку це кола та еліпси. Синіми (ближче до х\*) позначені лінії рівня з нижчим значенням, червоними – з вищим.

Зверніть увагу, що для поверхні, заданої рівнянням виду,визначає нормаль (перпендикуляр) до цієї поверхні. Також зверніть увагу, що хоч градієнт і показує у напрямку найбільшого збільшення функції, немає жодної гарантії, що у напрямку, зворотному до градієнта, можна знайти мінімум (приклад – ліва картинка).

## 5.2 Метод градієнтного спуску

Отже, ми навчилися за точкою x отримувати точкуіз меншим значенням функції f. Що заважає нам повторити це кілька разів? По суті, це градієнтний спуск: будуємо послідовність



Величинаназивається розміром кроку. Пару слів щодо вибору: якщо- дуже малі, то послідовність повільно змінюється, що робить алгоритм не дуже ефективним; якщо ждуже великі, то лінійне наближення стає поганим, а може, навіть і невірним.

Величина робочого кроку у напрямку градієнтаззалежить від величини градієнта, який заздалегідь врахувати складно, і від коефіцієнта пропорційності кроку h, за допомогою якого можна управляти ефективністю способу.

Пошук кожної нової точки складається з двох етапів:

1) оцінка градієнта f(х) Шляхом обчислення приватних похідних від f(х) По кожній змінній xj;

2) робочий крок по всіх змінних одночасно.

Розмір h сильно впливає ефективність методу. Більшою ефективністю має варіант методу, коли крок по кожній змінній визначається напрямними косинусами градієнта

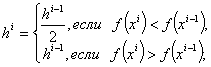
https://matica.org.ua/images/stories/MONMG/image303.gif

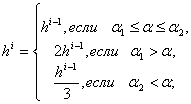
Де https://matica.org.ua/images/stories/MONMG/image304.gif

І тут величина робочого кроку залежить від величини модуля градієнта, і нею легше управляти. У районі оптимуму може виникати значне " нишпорення " , тому використовують різні алгоритми корекції h.

Найбільшого поширення набули такі алгоритми:

1)https://matica.org.ua/images/stories/MONMG/image305.gif(без корекції);

2)

3)

Деα- кут між градієнтами на попередньому та поточному кроці;α1іα2– задані граничні значення вибираються суб'єктивно (наприклад,α1=π/6,α2=π/3).

Вдалині від оптимуму напрямок градієнта змінюється мало, тому крок можна збільшити (другий вираз), поблизу оптимуму напрямок різко змінюється (кут між градієнтами f(x) великий), тому h скорочується (третє вираз).

Для оцінки приватних похідних використовуються різницеві методи:

•алгоритм із центральною пробою

https://matica.org.ua/images/stories/MONMG/image308.gif

•алгоритм із парними пробами

https://matica.org.ua/images/stories/MONMG/image309.gif

Де gi - пробний крок по i-й змінній, вибирається досить малим для різницевої оцінки похідної.

Перший алгоритм вимагає менших витрат у порівнянні з другим (зазвичай витрати виражаються кількістю обчислень критерію оптимальності), але дозволяє отримати[Рішення](http://matica.org.ua/sdelat-zakaz)менш точно, ніж другий, і це похибка залежить від величини пробного кроку.

Умовою закінчення пошуку може бути трохи модуля градієнта f(x), тобто |gradf(x)| <ε.

**Розглянемо приклад**. Потрібно знайти мінімум функціїзавершивши обчислення при похибці*ε*= 0.01, обравши початкове наближення х(0) = - 0,5 та у(0) = -1, коефіцієнт кроку h = 0.1.

**Рішення.**Необхідні початкові дані наведені за умови завдання. Для обчислень виберемо роботу з кроком "Без корекції" (h = const).

Знайдемо приватні похідні функції:

;

Отже,

;

Значить,



Змінні визначаються за формулами:

;

Результати обчислень

| № п/п | *x* | *y* |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | -0,5000 | -1,0000 | 7,3750 | -2,2500 | -8,0000 | 8,3104 |
| 2 | -0,2750 | -0,2000 | 1,6842 | -2,7731 | -4,8000 | 5,5435 |
| 3 | 0,0023 | 0,2800 | -0,9701 | -3,0000 | -2,8800 | 4,1586 |
| 4 | 0,3023 | 0,5680 | -2,5061 | -2,7258 | -1,7280 | 3,2274 |
| 5 | 0,5749 | 0,7408 | -3,4003 | -2,0085 | -1,0368 | 2,2603 |
| 6 | 0,7757 | 0,8445 | -3,8120 | -1,1947 | -0,6221 | 1,3469 |
| 7 | 0,8952 | 0,9067 | -3,9508 | -0,5958 | -0,3732 | 0,7031 |
| 8 | 0,9548 | 0,9440 | -3,9877 | -0,2651 | -0,2239 | 0,3471 |
| 9 | 0,9813 | 0,9664 | -3,9967 | -0,1111 | -0,1344 | 0,1744 |
| 10 | 0,9924 | 0,9798 | -3,9990 | -0,0453 | -0,0806 | 0,0925 |
| 11 | 0,9969 | 0,9879 | -3,9997 | -0,0183 | -0,0484 | 0,0517 |
| 12 | 0,9988 | 0,9927 | -3,9999 | -0,0073 | -0,0290 | 0,0299 |
| 13 | 0,9995 | 0,9956 | -4,0000 | -0,0029 | -0,0174 | 0,0177 |
| 14 | 0,9998 | 0,9974 | -4,0000 | -0,0012 | -0,0104 | 0,0105 |
| 15 | 0,9999 | 0,9984 | -4,0000 | -0,0005 | -0,0063 | 0,0063 |

В останній точці модуль градієнта менше заданої похибки (0,0063 < 0,01) тому пошук припиняється.

Відповідь (x\*, y\*) ≈ (0,9999; 0,9984) та f ≈ -4.

5.3 Метод якнайшвидшого спуску

Основним недоліком градієнтного методу є необхідність частого обчислення похідних від функції*f*(x). Цього недоліку позбавлений спосіб якнайшвидшого спуску, який полягає в наступному.

У поточній точці обчислюється, а потім у напрямку градієнта шукається. Практично це може бути здійснено будь-яким методом одновимірної оптимізації (пошук по одному напрямку - напрям градієнта), найчастіше використовується сканування до першого локального мінімуму за напрямком.

В результаті далеко від оптимуму ефективність методу підвищується, ми швидше потрапимо в район оптимуму, в околиці якого ефективність методу знижується через часту зміну напрямку пошуку і наближається до ефективності методу градієнта.

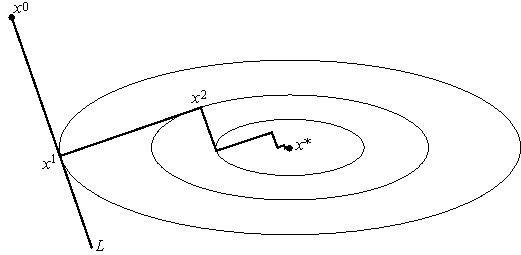
У ряді випадків можна підвищити швидкість виходу в район оптимуму пред'явленням невисоких вимог до точності пошуку min у напрямку (задається величиною h – кроком пошуку у напрямку). Умовою закінчення може бути трохи модуля градієнта функції*f*(x):. Можна також використовувати і небагато прирощень за змінними в результаті кроку, але тільки в тому випадку, якщо на цьому кроці ми «проскочили» оптимум, інакше може виявитися, що небагато кроку обумовлена ​​не близькістю до оптимуму, а трохи коефіцієнта пропорційності кроку h.

У ряді випадків використовують зменшення кроку пошуку оптимуму за напрямом після кожної зміни напрямку. Умовою закінчення пошуку у разі є досягнення заданої мінімальної величини кроку.

Метод якнайшвидшого спускуґрунтується на виборі кроку з наступного міркування. З точкибудемо рухатися в напрямку антиградієнта доти, доки не досягнемо мінімуму функціїна цьому напрямку, тобто на промені



Іншими словами,вибирається так, щоб наступна ітерація була точкою мінімуму функціїна промені L.



Такий варіант градієнтного методу називається методом якнайшвидшого спуску. Зауважимо, до речі, що в цьому методі напрямки сусідніх кроків ортогональні.

Метод якнайшвидшого спуску вимагає розв'язання кожному етапі завдання одномірної оптимізації. Практика показує, що цей метод часто вимагає менше операцій, ніж градієнтний метод з постійним кроком.

У загальній ситуації, проте, теоретична швидкість збіжності методу якнайшвидшого спуску не вища за швидкість збіжності градієнтного методу з постійним (оптимальним) кроком.

**Розглянемо приклад**. Мінімізувати функціюметодом якнайшвидшого спуску з точністю. Початкове наближення X0 (4-3).

**Рішення**. Знайдемо приватні похідні:,.

**1-а ітерація.**Як напрямок пошуку виберемо вектор градієнт у поточній точці:



Значення градієнта у точці X0:



Перевіримо критерій зупинки:



Маємо:



Обчислимо значення функції у початковій точці. Зробимо крок уздовж напрямку антиградієнта



Обчислимо значення функції у новій точці.



Знайдемо такий крок, щоб цільова функція досягала мінімуму вздовж цього напряму. З необхідної умови існування екстремуму функції:



Отримаємо крок:

Виконання цього кроку приведе до точки:



**2я ітерація**. Значення градієнта у точці X1:



Перевіримо критерій зупинки. Маємо:



Обчислимо значення функції у точці. Зробимо крок уздовж напрямку антиградієнта



Обчислимо значення функції у новій точці.



Знайдемо такий крок, щоб цільова функція досягала мінімуму вздовж цього напряму. З необхідної умови існування екстремуму функції:



Отримаємо крок:

Виконання цього кроку приведе до точки:



Далі продовжуємо робити ітерації доки не досягнемо заданої точності доки



# 6 Багатокритеріальна оптимізація

Теоретично багатокритеріальної оптимізації (МКО) вирішуються завдання прийняття рішень одночасно за декількома критеріями. Завдання МКО ставиться так: потрібно знайти числа, що задовольняють системі обмежень

,, (6.1)

для яких функції

,, (6.2)

досягають максимального значення.

Безліч точок, що задовольняють системі (6.1), утворює допустиму область. Елементи множининазиваються допустимими рішеннями чи альтернативами, а числові функції,– цільовими функціями, або критеріями, заданими на множині D. У формулюванні задачі (6.1)-(6.2) присутнійцільових функцій. Ці функції відображають безлічу безліч, Яке називається безліччю досяжності.

У векторній формі математичну модель МКО (6.1)-(6.2) можна записати так:

при. (6.3)

Тут- Вектор-функція аргументу.

Завдання вибору кращого варіанта передбачає наявність певної кількості конкуруючих варіантів, кожен з яких характеризується одним або декількома показниками якості, які можуть бути приватними критеріями. При цьому критерії характеризуються якісним напрямком:

– критерій якісно позитивний (має позитивний якісний напрямок), якщо якість варіанта тим більше, чим більше значення цього критерію;

– критерій якісно негативний (має негативне якісне напрям), якщо якість варіанта тим більше, що менше значення цього критерію.

Вперше проблема МКО виникла італійського економіста В.Парето в 1904 р. при математичному дослідженні товарного обміну. Надалі інтерес до проблеми МКО посилився у зв'язку з розробкою та використанням обчислювальної техніки, і вже пізніше стало ясно, що багатокритеріальні завдання виникають також у техніці, наприклад, при проектуванні складних технічних систем.

На відміну від завдань оптимізації з одним критерієм у МКО є невизначеність цілей. Справді, існування рішення, що максимізує кілька цільових функцій, є рідкісним винятком, тому з математичної точки зору завдання МКО є невизначеними і рішенням можливо лише компромісне рішення. Наприклад, при пошуку плану підприємства, що макимизує прибуток і мінімізує витрати очевидна неможливість досягнення обох цілей одночасно, тому що чим більше витрати, тим більше має бути продукції і тим більший прибуток.

З огляду на це теоретично МКО поняття оптимальності отримує різні тлумачення, і тому сама теорія містить три основних напрями:

1. Розробка концепції оптимальності.

2. Доказ існування рішення, оптимального у сенсі.

3. Розробка методів знаходження оптимального рішення.

## 6.1 Проблеми та класифікація методів вирішення задач багатокритеріальної оптимізації

При вирішенні завдань МКО доводиться вирішувати специфічні питання, пов'язані з невизначеністю цілей та несумірністю критеріїв. Перелічимо основні проблеми, що виникають розробки методів МКО.

1. Проблема нормалізації критеріїв, тобто приведення критеріїв до єдиного (безрозмірного) масштабу виміру.

2. Проблема вибору принципу оптимальності, тобто встановлення, у сенсі оптимальне рішення краще всіх інших решений.

3. Проблема обліку пріоритетів критеріїв, що виникає у тих випадках, коли з фізичного сенсу ясно, що деякі критерії мають пріоритет над іншими.

4. Проблема обчислення оптимуму завдання МКО. Йдеться про те, як використовувати методи лінійної, нелінійної, дискретної оптимізації для обчислення оптимуму завдань із певною специфікою.

При вирішенні багатокритеріального завдання часто виникає необхідність нормалізації (нормування) критеріїв, тобто приведення всіх критеріїв до єдиного масштабу та безрозмірного вигляду. Надалі вважатимемо, що це критерії неотрицательны, тобтодля всіх.

Найчастіше використовується заміна критеріїв їх безрозмірними відносними величинами:, де. Нормалізовані критерії мають дві важливі властивості: по-перше, вони є безрозмірними величинами, і, по-друге, вони задовольняють нерівностідля будь-кого. Ці властивості дозволяють порівнювати критерії між собою.

Основні методи, що застосовуються під час вирішення завдань МКО, представлені малюнку нижче.

**Методи вирішення**

**багатокритеріальних завдань**

Інтерактивні

Лексикографічна оптимізація

Зведення до однокритеріальних

Метод аналізу ієрархій

Метод ефективних множин

Метод

поступок

Метод головного критерію

Метод згортки

Метод цільового програмування

Адитивні

Мультиплікативні

Максимінні

## 6.2 Метод адитивної згортки

Методи, засновані на згортанні критеріїв полягають у тому, що замістьприватних критеріїврозглядається один скалярний критерій, отриманий шляхом поєднання приватних критеріїв.

Нехай критерії співмірні, наприклад, нормовані та визначено вектор вагових коефіцієнтів критеріїв, що характеризують важливість відповідного критерію Це означає що, якщо критеріймає пріоритет над критерієм. При цьому

,.

Для адитивного методу будується нова цільова функція

 (6.4)

та вирішується завдання оптимізації скалярного критерію

за умови.

**Розглянемо завдання**. Ринок пропонує ряд моделей автомобілів, що відрізняються вартістю С та максимальною швидкістю

|  |  |  |
| --- | --- | --- |
| № моделі |  |  |
| 1 | 40 | 1600 |
| 2 | 50 | 2500 |
| 3 | 60 | 3600 |
| 4 | 70 | 4900 |
| 5 | 80 | 6400 |
| 6 | 90 | 8100 |
| 7 | 100 | 10000 |
| 8 | 110 | 12100 |
| 9 | 120 | 14400 |
| 10 | 130 | 16900 |
| 11 | 140 | 19600 |
| 12 | 150 | 22500 |
| 13 | 160 | 25600 |

Необхідно вибрати оптимальний (максимальна швидкість за мінімальних витрат) варіант за умови, що пріоритети критеріїв рівні: 0,6 (вартість) та 0,4 (максимальна швидкість).

**Рішення**. Насамперед зазначимо, що вихідні критерії якісно позитивний (швидкість) та негативний (вартість). На першому етапі необхідно синхронізувати цілі. Це можна зробити, розглянувши «протилежний» до одного з критеріїв. Наприклад, для швидкості можна розглянути «загальмованість». Вартість можна перетворити на «економію». Різниця між цими варіантами зводиться до того яке в результаті завдання ми вирішуватимемо мінімізації або максимізації. Виберемо варіант із «загальмованістю».

Далі необхідно нормалізувати критерії, оскільки їх діапазони значень суттєво відрізняються. Для цього необхідно розділити всі критерії на відповідні максимальні значення цих критеріїв. У нашому випадку 160 для швидкості та 25600 для вартості. Отримані таким чином нормалізовані значенняіпідставляємо у вираз (6.4). Дані розрахунків наведено у таблиці нижче.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № моделі |  |  |  |  |  |  |  |  |
| 1 | 40 | 1600 | 120 | 0,0625 | 1 | 0,0375 | 0,4 | 0,4375 |
| 2 | 50 | 2500 | 110 | 0,097656 | 0,916667 | 0,058594 | 0,366667 | 0,42526 |
| 3 | 60 | 3600 | 100 | 0,140625 | 0,833333 | 0,084375 | 0,333333 | 0,417708 |
| 4 | 70 | 4900 | 90 | 0,191406 | 0,75 | 0,114844 | 0,3 | 0,414844 |
| 5 | 80 | 6400 | 80 | 0,25 | 0,666667 | 0,15 | 0,266667 | 0,416667 |
| 6 | 90 | 8100 | 70 | 0,316406 | 0,583333 | 0,189844 | 0,233333 | 0,423177 |
| 7 | 100 | 10000 | 60 | 0,390625 | 0,5 | 0,234375 | 0,2 | 0,434375 |
| 8 | 110 | 12100 | 50 | 0,472656 | 0,416667 | 0,283594 | 0,166667 | 0,45026 |
| 9 | 120 | 14400 | 40 | 0,5625 | 0,333333 | 0,3375 | 0,133333 | 0,470833 |
| 10 | 130 | 16900 | 30 | 0,660156 | 0,25 | 0,396094 | 0,1 | 0,496094 |
| 11 | 140 | 19600 | 20 | 0,765625 | 0,166667 | 0,459375 | 0,066667 | 0,526042 |
| 12 | 150 | 22500 | 10 | 0,878906 | 0,083333 | 0,527344 | 0,033333 | 0,560677 |
| 13 | 160 | 25600 | 0 | 1 | 0 | 0,6 | 0 | 0,6 |

Як бачимо з таблиці екстремум цільової функції (0,414) відповідає моделі 4. Її потрібно брати.

Варто зазначити, що в задачах багатокритеріальної оптимізації ми отримуємо те рішення, яке максимально узгоджується з нашою картиною світу, яка визначається пріоритетами критеріїв. Якщо в розглянутій задачі ми, наприклад, приймемо, що пріоритети критеріїв дорівнюватимуть 0.5, то рішенням вже буде варіант 8.

## 6.3 Метод мультиплікативної згортки

Для мультиплікативного методу підхід до рішення аналогічний, тільки цільова функція має вигляд

, (6.5)

причому.

Відмінна особливість мультиплікативної згортки в тому, що для неї не потрібно проводити нормалізацію критеріїв.

**Розглянемо рішення**Завдання про вибір автомобіля методом мультиплікативної згортки.

Т.к. проводити нормалізацію критеріїв немає необхідності, на першому етапі, як і у разі адитивного згортки, замінимо якісно позитивний критерій швидкість на якісно негативний. Потім зведемо значення критеріїв у відповідному ступеніі розмножимо їх. Результати розрахунків наведено у таблиці нижче.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № моделі |  |  |  |  |  |  |
| 1 | 40 | 1600 | 120 | 6,786916 | 333,0213 | 2260,188 |
| 2 | 50 | 2500 | 110 | 6,554764 | 435,2753 | 2853,127 |
| 3 | 60 | 3600 | 100 | 6,309573 | 541,7283 | 3418,074 |
| 4 | 70 | 4900 | 90 | 6,049187 | 651,8049 | 3942,89 |
| 5 | 80 | 6400 | 80 | 5,7708 | 765,082 | 4415,135 |
| 6 | 90 | 8100 | 70 | 5,470654 | 881,2335 | 4820,923 |
| 7 | 100 | 10000 | 60 | 5,143521 | 1000 | 5143,521 |
| 8 | 110 | 12100 | 50 | 4,781762 | 1121,169 | 5361,166 |
| 9 | 120 | 14400 | 40 | 4,373448 | 1244,565 | 5443,04 |
| 10 | 130 | 16900 | 30 | 3,89806 | 1370,036 | 5340,483 |
| 11 | 140 | 19600 | 20 | 3,314454 | 1497,455 | 4963,244 |
| 12 | 150 | 22500 | 10 | 2,511886 | 1626,708 | 4086,105 |
| 13 | 160 | 25600 | 0 | 0 | 1757,697 | 0 |

У цьому випадку бачимо, що екстремум (5443,04) цільової функції відповідає моделі №9.

## 6.4 Метод цільового програмування

Назва цієї групи методів пов'язана з тим, що особа, яка приймає рішення (ЛПР), задає певні цілі.для кожного критерію. Завдання МКО у разі перетворюється на завдання мінімізації суми відхилень з певним показником:

, при XПроD, (6.6)

де- Деякі вагові коефіцієнти, що характеризують важливість того чи іншого критерію.

Завдання (6.6) можна конкретизувати залежно від значень параметрата заданих цілей. Зокрема, приіотримаємо задачу мінімізації суми квадратів відхилень:

при XПроD,

в якій мінімізується евклідова відстань від множини досяжності F до «абсолютного максимуму»у просторі критеріїв. Тут.

**Розглянемо рішення**Завдання про вибір автомобіля методом цільового програмування.

Насамперед проведемо нормалізацію критеріїв розділивши їх на відповідне максимальне значення. Далі визначимося з цілями. Очевидно, що для швидкості метою буде максимальне значення 1, а для мінімальної вартості — 0. Подальші розрахунки наведені в таблиці нижче.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № моделі |  |  |  |  |  |  |  |
| 1 | 40 | 1600 | 0,25 | 0,0625 | 0,225 | 0,002344 | 0,476806 |
| 2 | 50 | 2500 | 0,3125 | 0,097656 | 0,189063 | 0,005722 | 0,441344 |
| 3 | 60 | 3600 | 0,375 | 0,140625 | 0,15625 | 0,011865 | 0,410019 |
| 4 | 70 | 4900 | 0,4375 | 0,191406 | 0,126563 | 0,021982 | 0,385414 |
| 5 | 80 | 6400 | 0,5 | 0,25 | 0,1 | 0,0375 | 0,37081 |
| 6 | 90 | 8100 | 0,5625 | 0,316406 | 0,076563 | 0,060068 | 0,369635 |
| 7 | 100 | 10000 | 0,625 | 0,390625 | 0,05625 | 0,091553 | 0,384451 |
| 8 | 110 | 12100 | 0,6875 | 0,472656 | 0,039063 | 0,134042 | 0,416059 |
| 9 | 120 | 14400 | 0,75 | 0,5625 | 0,025 | 0,189844 | 0,463512 |
| 10 | 130 | 16900 | 0,8125 | 0,660156 | 0,014063 | 0,261484 | 0,524925 |
| 11 | 140 | 19600 | 0,875 | 0,765625 | 0,00625 | 0,351709 | 0,598297 |
| 12 | 150 | 22500 | 0,9375 | 0,878906 | 0,001563 | 0,463486 | 0,681944 |
| 13 | 160 | 25600 | 1 | 1 | 0 | 0,6 | 0,774597 |

Як бачимо, в даному випадку оптимальне рішення відповідає моделі автомобіля під номером 6.

## 6.5 Метод аналізу ієрархій

У світі ви постійно стикаєтеся з необхідністю вибору. Вибір може бути найпростішим. Вибір може бути обмежений наявними ресурсами. Вибір може бути складним, яке наслідки неоднозначними. Наприклад, чим опалювати квартиру? Газом чи електрикою? Газ дешевший. Але газова колонка давня і вже на дихає. Коефіцієнт корисної дії в неї лише на рівні 0,4 – 0,5. Це означає, що більше половини тепла вилітає у трубу. Та й коли вона включається, то гуде, скрипить, тріщить так, ніби зараз розвалиться. А ще газова служба розбавила газ і його енергоємність уже значно нижча за стандартне значення. А днями у вас поставили багатотарифний лічильник електроенергії і тепер ночами вартість електроенергії значно нижча за денну вартість.

Для вирішення подібних багатокритеріальних завдань розроблено спеціальні методи і один з них – це метод аналізу ієрархій або МАІ. Його розробив американський математик Томас Сааті.

Слід зазначити, що МАІ не дає якогось «правильного» рішення, а дозволяє знайти таку альтернативу, яка найкраще узгоджується з розумінням суті проблеми та вимогами до її вирішення. Тобто рішення залежить від того наскільки добре ви розумієте суть завдання. Якщо ви зміните свої міркування, рішення може також змінитись.

Суть методу у проходженні наступних етапів:

1) Побудова якісної моделі проблеми як ієрархії

2) Визначення пріоритетів всіх елементів ієрархії за допомогою методу парних порівнянь

3) Перевірка суджень на узгодженість

4) Синтез глобальних пріоритетів альтернатив

5) Прийняття рішення на основі одержаних результатів

Розглянемо таку задачу. Допустимо ви вирішили заробити продаючи зволожуючі креми, але в самих кремах ви зовсім не знаєтеся. Зібрати інформацію про плюси і мінуси того чи іншого крему ви можете без проблем, але ця інформація є набором різнорідних критеріїв, які неможливо прямо порівняти. Ви ставите собі за мету вибрати найкращий варіант.

Пошукавши в інтернеті статті про зволожуючі креми, ви склали для себе наступний список альтернатив

Крем для обличчя L'Oreal Revitalift

Крем для обличчя нічний

Крем для обличчя Чорні Перли

Переглянувши форуми та почитавши відгуки ви визначили для себе критерії, які, на ваш погляд, важливі для вирішення цього завдання. Це: Ціна, Факти появи алергічних реакцій, Ефективність зволоження та Аромат.

Якщо з ціною все ясно, то інші 3 критерії вимагають пояснення. Аналіз відгуків на форумах показав, що крем L'Oreal чудово зволожує шкіру та має чудовий аромат, але іноді викликає алергічні реакції. За іншими кремами фактів алергії не виявлено, але аромат у NIVEA не сказати, щоб якийсь. А чорні перли взагалі найдешевші.

Подумавши ви вирішили оцінити ефективність зволоження за шкалою від 1 до 10, алергію відзначити як факт так/ні, а аромат оцінити як приємний, ніякий і не дратує.

Так у вас з'явилися вихідні дані, необхідні для вирішення задачі.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ціна | Алергенність | Зволоження | Аромат |
| Крем для обличчя L'Oreal Revitalift Лазер Х3 | 270 | + | 10 | Приємний |
| Крем для обличчя нічний | 74 | - | 8 | ніякий |
| Крем для обличчя Чорні Перли | 64 | - | 6 | Не дратівливий |

Розглянемо перший етап МАІ: побудова моделі проблеми як ієрархії.

Ієрархічна структура — це графічне уявлення проблеми як перевернутого дерева, де кожен елемент, крім найвищого, залежить від однієї чи більше вище розташованих елементів. Часто у різних організаціях розподіл повноважень між співробітниками організовано в ієрархічній формі. Так у вашій школі є директор, у нього в підпорядкуванні завучі, далі йдуть викладачі та учні.



У разі верхній рівень ієрархії (він же наша мета) це крем. На другому рівні розташовані критерії, які на наш погляд визначають вирішення задачі. На третьому рівні розташовані альтернативи.

Наступний етап МАІ: визначення локальних пріоритетів усіх елементів ієрархії. Іншими словами ми повинні визначити наскільки одні критерії важливі в порівнянні з іншими, а також як пріоритет альтернатив по кожному з критеріїв. В результаті ми отримаємо кількісні оцінки важливості критеріїв та альтернатив.

Цей етап заснований на про парних порівняннях. Ми будемо брати пару критеріїв чи альтернатив та порівнювати їх між собою. При цьому ставитимемо питання «на скільки перший критерій важливіший за другий?» «на скільки перша альтернатива краща чи гірша за другу?» і т.п.

За результатами порівнянь ми побудуємо звані матриці парних порівнянь.

Тут з'являється аспект. Якщо оцінити наші креми за критерієм вартості більш-менш зрозуміло як, то як їх оцінити за критеріями алергія, зволоження та аромат? Беручи вартість, ми можемо розділити одну на іншу і отримати їхнє відношення. А як знайти кількісну оцінку відношення аромату приємного до недратівливого?

Звичайно нам знадобиться шкала, за якою можна будувати наші оцінки.

Така шкала була вигадана. Порівнюючи два об'єкти між собою (чи то критерії чи альтернативи) ми повинні поставити число в інтервалі від 1 до 9 якщо перший об'єкт кращий за другий і зворотне значення якщо перший об'єкт гірший за другий.

|  |  |  |
| --- | --- | --- |
| Ступінь значущості | Визначення | Пояснення |
| 1 | Однакова значимість | Дві дії роблять однаковий внесок у досягнення мети |
| 3 | Деяке переважання значущості однієї дії над іншим | Існують міркування на користь переваги однієї з дій, проте ці міркування недостатньо переконливі |
| 5 | Істотна чи сильна значимість | Є надійні дані або логічні міркування для того, щоб показати перевагу однієї з дій |
| 7 | Очевидне чи дуже сильне значення | Переконливе свідчення на користь однієї дії перед іншою |
| 9 | Абсолютна значимість | Свідоцтва на користь переваги однієї дії перед іншою переконливі |
| 2, 4, 6, 8 | Проміжні значення між двома сусідніми судженнями | Ситуація, коли необхідне компромісне рішення |

Раніше наголошувалося, що метод МАІ дозволяє вирішити поставлене завдання у рамках нашого уявлення про неї. То наскільки коректно ми підійдемо до кількісної оцінки парних порівнянь і визначимо те рішення, яке ми отримаємо надалі. Це найвідповідальніший етап.

Перша матриця демонструє оцінку важливості критеріїв між собою. Вирішили, що наявність алергічних реакцій важливіше проти іншими критеріями. Так алергія має переважну перевагу над критерієм аромат. Тут була поставлена ​​оцінка 9. Відповідно в клітині ставлення Аромата до Алергії стоїть обернена величина 1/9. У ході порівняння Алергії з ефектом зволоження, було зроблено висновок, що алергія важлива, але і зволоження теж має вагу. Тому їх відношення виразилося оцінкою 3. Подібним чином були зроблені всі парні порівняння. Якщо відповідній клітині з'являлася деяка оцінка, то клітині симетричної щодо діагоналі ставилася зворотна величина.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Ціна | Алергенність | Зволоження | Аромат |
| Ціна | 1 | 1/7 | 1/3 | 3 |
| Алергенність | 7 | 1 | 3 | 9 |
| Зволоження | 3 | 1/3 | 1 | 6 |
| Аромат | 1/3 | 1/9 | 1/6 | 1 |

Далі були заповнені матриці парних порівнянь альтернатив по кожному з критеріїв.

|  |  |  |  |
| --- | --- | --- | --- |
| Ціна | L'Oreal | NIVEA | Чорні перли |
| Крем для обличчя L'Oreal Revitalift Лазер Х3 | 1 | 1/8 | 1/9 |
| Крем для обличчя нічний | 8 | 1 | 1/3 |
| Крем для обличчя Чорні Перли | 9 | 3 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Алергенність | L'Oreal | NIVEA | Чорні перли |
| Крем для обличчя L'Oreal Revitalift Лазер Х3 | 1 | 1/9 | 1/9 |
| Крем для обличчя нічний | 9 | 1 | 1 |
| Крем для обличчя Чорні Перли | 9 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Зволоження | L'Oreal | NIVEA | Чорні перли |
| Крем для обличчя L'Oreal Revitalift Лазер Х3 | 1 | 3 | 6 |
| Крем для обличчя нічний | 1/3 | 1 | 3 |
| Крем для обличчя Чорні Перли | 1/6 | 1/3 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Аромат | L'Oreal | NIVEA | Чорні перли |
| Крем для обличчя L'Oreal Revitalift Лазер Х3 | 1 | 3 | 2 |
| Крем для обличчя нічний | 1/3 | 1 | 1/2 |
| Крем для обличчя Чорні Перли | 1/2 | 2 | 1 |

Після складання матриць парних порівнянь необхідно вирахувати локальні пріоритети. Як оцінка відповідних пріоритетів приймаються значення координат власних векторів матриць. А вони, своєю чергою, визначаються як середнє геометричне значення оцінок у відповідних рядках.

На прикладі матриці порівняння критеріїв для критерію «Ціна» ми отримаємо:

.

Для альтернативи L'Oreal в матриці порівняння за критерієм ціна ми відповідно отримаємо:

.

Таким чином, ми отримаємо координати деяких векторів, які якось характеризують важливість відповідних альтернатив.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Власні вектори матриць порівнянь | | | | |
| Критерії | Ціна | Алергенність | Зволоження | Аромат |
| 0,615 | 0,240375 | 0,23112 | 2,620741 | 1,817121 |
| 3,708 | 1,386723 | 2,080084 | 1 | 0,550321 |
| 1,565 | 3 | 2,080084 | 0,381571 | 1 |
| 0,280 |  |  |  |  |

Проблема в тому, що всі ці вектори отримані, умовно кажучи, у різних масштабах, тому для коректного їх порівняння необхідно провести їхнє нормування. Для цього обчислюємо суму всіх координат для відповідного вектора та ділимо кожну координату на отриману суму. Якщо ви все зробили правильно, то сума координат кожного вектора дорівнюватиме 1.

Нормовані значення власних векторів, які тепер можна розглядати як чисельні значення локальних векторів пріоритетів, представлені нижче.

Далі необхідно здійснити перевірку узгодженості наших суджень. Тобто треба перевірити, чи не суперечать один одному наші судження. Раптом ми виставляли їх особливо не замислюючись? Раптом ми сказали, що a > b > c > a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Нормовані значення власних векторів матриць порівнянь | | | | |
| Критерії | Ціна | Алергенність | Зволоження | Аромат |
| 0,100 | 0,051949 | 0,052632 | 0,654807 | 0,539615 |
| 0,601 | 0,299696 | 0,473684 | 0,249856 | 0,163424 |
| 0,254 | 0,648355 | 0,473684 | 0,095338 | 0,296961 |
| 0,045 |  |  |  |  |

Математика дозволяє підрахувати кількісно, ​​на скільки ми безвідповідально підійшли до питання, коли виставляли парні оцінки.

Розглянемо матрицю критеріїв. Нам потрібно порахувати три величини. Це число власне число, індекс узгодженості (ІВ) та відношення узгодженості (ОС). Ці величини визначають узгодженість матриці.

Для розрахункунеобхідно підсумувати кожен стовпець матриці. Далі суму першого стовпця необхідно помножити першу координату локального вектора пріоритетів. Суму другого стовпця другого компоненту і т.д. Отримані значення слід скласти. Якщо ви все зробили правильно, то величинабуде більше або дорівнює порядку матриці.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ціна | Алергенність | Зволоження | Аромат | Локальний вектор пріоритетів |  |  |
| Ціна | 1 | 1/7 | 1/3 | 3 | 0,100 |  |  |
| Алергенність | 7 | 1 | 3 | 9 | 0,601 |  |  |
| Зволоження | 3 | 1/3 | 1 | 6 | 0,254 |  |  |
| Аромат | 1/3 | 1/9 | 1/6 | 1 | 0,045 |  |  |
| Σ | 11,33 | 1,59 | 4,50 | 19,00 |  |  |  |
|  | | | | | |  | 4,089 |
|  | | | | | | ІС | 0,030 |
|  |  |  |  |  |  | ОС | 0,033 |

Знаючи величинуми можемо визначити ІС за виразом. Ця величина характеризує узгодженість матриці, але у абсолютних одиницях. Тобто. ми не можемо сказати багато це чи мало. Потрібно порівняти цю величину з чимось нам зрозумілим.

Такою величиною є довільна узгодженість (СС). Вона таки показує узгодженість матриці, у якій оцінки виставлені випадковим чином. СС залежить від порядку матриці та є табличною величиною. Для матриці 4 СС становить 0.9.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Розмір матриці | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Випадкова узгодженість | 0 | 0 | 0,58 | 0,9 | 1,12 | 1,24 | 1,32 | 1,41 | 1,45 | 1,49 |

Якщо ми тепер знайдемо відношення ІС до СС, то отримаємо величину відношення узгодженості яка і характеризує наскільки наші оцінки відрізняються від випадкових.

Якщо ОС лежить у межах 0-15%, то все гаразд. Якщо більше 20% – треба переглядати наші оцінки. У разі ОС становило ~3%.

Аналогічним чином проводиться перевірка узгодженості матриць порівнянь, що залишилися. Якщо скрізь узгодженість у межах норми, можна переходити до обчислення глобальних пріоритетів. Якщо ж десь ОС занадто високо, необхідно переглянути оцінки у відповідних матрицях парних порівнянь.

|  |  |  |
| --- | --- | --- |
|  | ІС | ОС |
| Критерії | 0,030 | 0,033 |
| Ціна | 0,054 | 0,093 |
| Алергенність | 0,000 | 0,000 |
| Зволоження | 0,009 | 0,016 |
| Аромат | 0,005 | 0,008 |

У нашому випадку ОС не перевищує 10%, тому переходимо до останнього етапу – синтезу глобальних пріоритетів.

Тепер у нас є все для того, щоб порахувати вагу кожної альтернативи по відношенню до глобальної мети та відповісти на запитання, що краще?

Формально зібравши разом вектори локальних пріоритетів альтернатив, ми отримаємо якусь матрицю.

|  |  |  |  |
| --- | --- | --- | --- |
| 0,052 | 0,053 | 0,655 | 0,540 |
| 0,300 | 0,474 | 0,250 | 0,163 |
| 0,648 | 0,474 | 0,095 | 0,297 |

Для отримання глобальних пріоритетів цю матрицю необхідно помножити на вектор локальних пріоритетів критеріїв. По кожній альтернативі необхідно помножити пріоритет альтернативи за критерієм на відповідний пріоритет критерію. Отримані твори потрібно скласти.



В результаті ми отримаємо розв'язання задачі. Розрахований вектор показує кількісні оцінки кожної альтернативи у межах обраних критеріїв. Так бачимо, що критерій алергія має переважну перевагу проти іншими. Його вага 0,601. Через це крем L`Oreal отримав найменший світовий пріоритет 0,227. З іншого боку плюси та мінуси кремів NIVEA та ЧЖ компенсують один одного та їх пріоритети практично рівні.

Ще раз слід зазначити, що отримане рішення залежить від проведених вами оцінок і суджень. Якщо хтось інший візьметься за вирішення цього завдання, то можливо отримає інше рішення. Воно відображатиме саме його картину цінностей.

Хоча це завдання носить швидше жартівливий і навчальний характер, вирішення реальних завдань здійснюється так само. Коли перед вами стоятиме вибір куди піти, що зробити, що придбати? Згадайте про цей спосіб.

# 7 Апроксимація

При обробці експериментальних даних часто виникає необхідність апроксимувати їх лінійною функцією.

Апроксимацією(Наближенням) функціїназивається знаходження такої функції (аппроксимуючої функції), яка була б близька до заданої. Критерії близькості функцій можуть бути різні.

Якщо функціяє дискретним набором точок, апроксимацію називають точковою або дискретною.

Якщо апроксимація проводиться на безперервній множині точок (відрізку), апроксимація називається безперервною або інтегральною. Прикладом такої апроксимації може бути розкладання функції ряд Тейлора, тобто заміна деякої функції статечним многочленом.

## 7.1 Апроксимація лінійною функцією

Будь-яка лінійна функція може бути записана рівнянням



Апроксимація полягає у відшуканні коефіцієнтів a і b рівняння таких, щоб усі експериментальні точки лежали найближче до апроксимуючої прямої.

З цією метою найчастіше використовується метод найменших квадратів (МНК), суть якого полягає в наступному: сума квадратів відхилень значення точки від апроксимуючої точки набуває мінімального значення:



Вирішення поставленого завдання зводиться до знаходження екстремуму зазначеної функції двох змінних. З цією метою знаходимо приватні похідні функції за коефіцієнтами a і b і прирівнюємо їх до нуля.



Вирішуємо отриману систему рівнянь

 (7.1)

Визначаємо значення коефіцієнтів



## 7.2 Апроксимація іншими залежностями

Для апроксимації експериментальних точокпараболоюслід знайти мінімум функції 3х змінних:



Для цього необхідно вирішити систему рівнянь:



Припустимо, розташування експериментальних точок нагадує гіперболу. Тоді щоб знайти коефіцієнти а, b кращої гіперболи, потрібно знайти мінімум функції:



Бажаючі можуть провести докладні обчислення та дійти схожої системи:



З формального погляду вона виходить із «лінійної» системи (7.1) заміною «ікса» на.

Якщо є підстави припускати, що точкирозташовуються на логарифмічній кривій, то пошуку оптимальних значень а, b знаходимо мінімум функції:



Формально в системі (7.1) потрібно замінити х на ln(x):



З експоненційною залежністю(b > 0) ситуація трохи складніша. Щоб звести справу до лінійного випадку, прологарифмуємо функцію та скористаємося властивостями логарифму:



Тепер, зіставляючи отриману функцію з лінійною функцією, приходимо до висновку, що в системі (7.1) необхіднозамінити на, а- На. Для зручності позначимо:



Зверніть увагу, що система дозволяється щодоі, і тому після знаходження коріння потрібно не забути знайти сам коефіцієнт.

# 8 Інтерполяція

Інтерполяція – спосіб знаходження проміжних значень величини наявного дискретного набору відомих значень.

Нехай у ході експерименту при зміні вхідної величини х (x0, x1, x2,..., xn) отримано значення функції y=f(x) (y0, y1, y2.....yn)

Експериментальні дані

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x*0 | *x*1 | *x*2 | … | *x*n-1 | *x*n |
| *y*0 | *y*1 | *y*2 | … | *y*n-1 | *y*n |

Інтерполяцію функцій застосовують у разі, коли потрібно знайти значення функціїпри значенні аргументу, що належить інтервалу, але не збігається за значенням з жодним значенням, наведеним у таблиці вище.

Це завдання, зокрема інтерполяція функцій, часто зустрічається при обмеженості можливостей під час проведення експерименту. Зокрема через дорожнечу та трудомісткість проведення експерименту розмір вибірки (x0, x1, x2,..., xn) може бути досить малий.

При цьому у багатьох випадках аналітичний вираз функціїневідомо та отримати його за таблицею її значень (табл. 1) у більшості випадків неможливо. Тому замість неї будують іншу функцію, яка легко обчислюється і має ту ж саму таблицю значень (збігається з нею в точках x0, x1, x2,..., xn), що і, тобто.

 (8.1)

де i = 0, 1, 2, ..., n.

Знаходження наближеної функції називається інтерполяцією, а точки x0, x1, x2,..., xn – вузлами інтерполяції.

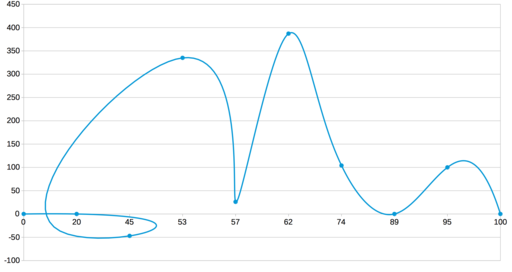
Інтерполюючу функцію шукають у вигляді полінома n ступеня.

Для кожного набору точок є лише один інтерполяційний багаточлен, ступеня не більший за n. Однозначно певний многочлен може бути представлений у різних видах.

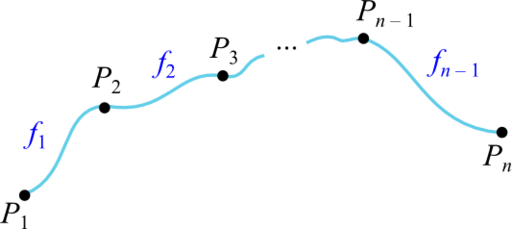
Графічно завдання інтерполювання полягає в тому, щоб побудувати таку інтерполюючу функцію, яка проходила б через усі вузли інтерполювання:



У той же час потрібно бути обережним, щоб не вийшло такого:



Є безліч способів інтерполяції, але вони можуть бути зведені до того, що треба знайти n–1 функцію для розрахунку проміжних точок на відповідних сегментах. При цьому задані точки обов'язково мають бути обчислювані через відповідні функції. На основі цього і може бути побудований графік:



Функціїможуть бути різними, але найчастіше використовують поліноми деякою мірою. У цьому випадку підсумкова інтерполююча функція (шматково задана на проміжках, обмежених точками Pi) називається сплайном.

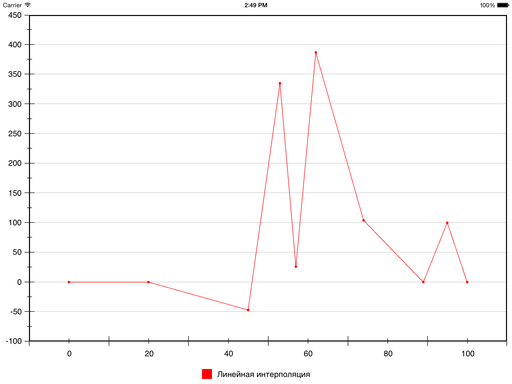
У різних інструментах для побудови графіків – редакторів та бібліотек – завдання «красивої інтерполяції» вирішено по-різному.

Найпростіший приклад - лінійна інтерполяція, в якій використовуються поліноми першого ступеня, а в результаті виходить ламана, яка сполучає задані точки.

Давайте додамо трохи конкретики. Ось набір точок:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 20 | 45 | 53 | 57 | 62 | 74 | 89 | 95 | 100 |
|  | 0 | 0 | -47 | 335 | 26 | 387 | 104 | 0 | 100 | 0 |

Результат лінійної інтерполяції цих точок виглядає так:



Але іноді хочеться отримати в результаті гладку криву.

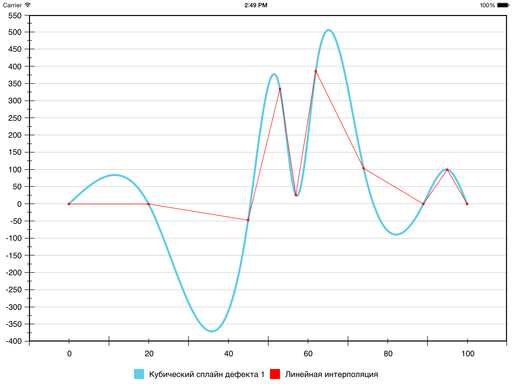
Що таке гладкість? Побутова відповідь: відсутність гострих кутів. Математичний: безперервність похідних. При цьому в математиці гладкість має порядок, рівний номеру останньої безперервної похідної, і область, де ця безперервність зберігається. Тобто, якщо функція має гладкість порядку 1 на відрізку [a;b], це означає, що на [a;b] вона має безперервну першу похідну, а друга похідна вже терпить розрив в якихось точках.

Сплайн в контексті гладкості має поняття дефекту. Дефект сплайну – це різниця між його ступенем та його гладкістю. Ступінь сплайну - це максимальна міра використаних у ньому поліномів.

Важливо відзначити, що «небезпечними» точками сплайну (в яких може порушитися гладкість) є якраз Pi, тобто точки зчленування сегментів, в яких відбувається перехід від одного полінома до іншого. Всі інші точки «безпечні», адже поліном на області його визначення не має проблем із безперервністю похідних.

Щоб досягти гладкої інтерполяції, потрібно підвищити рівень поліномів і підібрати їх коефіцієнти так, щоб у граничних точках зберігалася безперервність похідних.

Традиційно для вирішення такого завдання використовують поліноми третього ступеня та домагаються безперервності першої та другої похідної. Те, що виходить, називають кубічним сплайном дефекту 1. Ось як він виглядає для наших даних:

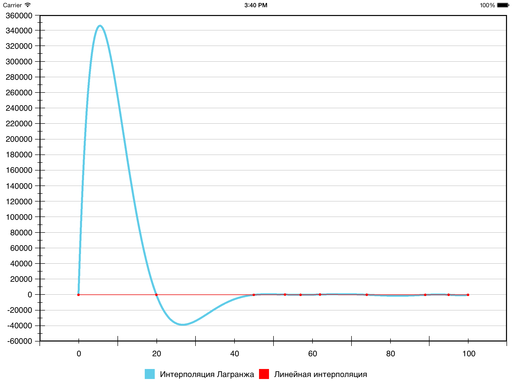


Крива, справді, гладка. Але якщо припустити, що це графік деякого процесу чи явища, який потрібно показати зацікавленій особі, то такий метод, швидше за все, не підходить. Проблема у хибних екстремумах. З'явилися вони через занадто сильне викривлення, яке покликане забезпечити гладкість інтерполяційної функції. Але глядачеві така поведінка зовсім не до речі, адже він виявляється обдурений щодо пікових значень функції. А заради наочної візуалізації цих значень, власне, все й починалося.

Тож треба шукати інші рішення.

Інше традиційне рішення, крім кубічних сплайн дефекту 1 - поліноми Лагранжа. Це поліноми ступеня n-1, що приймають задані значення заданих точках. Тобто членування на сегменти не відбувається, вся послідовність описується одним полиномом.

Але ось що виходить:



Гладкість, звичайно, є, але наочність постраждала дуже сильно.

Існують методи інтерполяції, що дозволяють досягти і гладкості та порівняно малої похибки, але їх розгляд виходить за рамки даного курсу. Розглянемо деякі основні методи.

## 8.1 Канонічний поліном

Вид канонічного полінома ступеня n

 (8.2)

Вибір многочлена ступеня n заснований на тому, що через n+1 точку проходить єдина крива ступеня n. Підставивши (8.2) (8.1), отримаємо систему лінійних алгебраїчних рівнянь (8.3)



Вирішуючи цю систему лінійних рівнянь алгебри, знайдемо коефіцієнти інтерполяційного полінома.

## 8.2 Лінійна інтерполяція

Лінійна інтерполяція – найпростіший вид інтерполяції, що часто використовується. Вона у тому, що задані точки з координатами xi, yi при i=0, 1, 2, ... n з'єднуються прямолінійними відрізками, а функцію y(x) можна приблизно у вигляді ламаною.

Рівняння кожного відрізка ламаною у випадку різні. Оскільки є n інтервалів (xi-1, xi), то для кожного з них як рівняння інтерполяційного багаточлена використовується рівняння прямої, що проходить через дві точки: для i-го інтервалу можна написати рівняння прямої, що проходить через точки (xi-1, yi -1) та (xi, yi),



Звідси





Розглянемо приклад.

Експериментальні дані

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x* | 0 | 2 | 3 | 3,5 |
| *y* | -1 | 0,2 | 0,5 | 0,8 |

Знайти значення функції за x = 1 і x = 3,2.

Рішення.

Крапка x = 1 належить першому локальному відрізку [0, 2], тобто. i=1 і, отже, за наведеними вище формулами:



Крапка x = 3,2 належить третьому інтервалу [3, 3,5], тобто. i=3 і, отже:



## 8.3 Інтерполяція кубічними сплайнами

Деяка функція f(x) задана на відрізку [a,b], розбитому на частині [xi − 1,xi], a = x0 < x1 < ... < xN = b. Кубічним сплайном дефекту 1 називається функція S(x), яка:

— на кожному відрізку [xi − 1,xi] є багаточлен ступеня не вище третього;

- має безперервні першу та другу похідні на всьому відрізку [a, b];

— у точках xi виконується рівність S(xi) = f(xi), тобто сплайн S(x) інтерполює функцію f у точках xi.

Для однозначного завдання сплайну перерахованих умов недостатньо, для побудови сплайну необхідно накласти якісь додаткові вимоги.

Природним кубічним сплайном називається кубічний сплайн, який також задовольняє граничним умовам виду:

*S*''(a) = S''(b) = 0.

Для будь-якої функції f та будь-якого розбиття відрізка [a,b] існує рівно один природний сплайн S(x), що задовольняє переліченим вище умовам.

Позначимо: hi = xi − xi − 1

На кожному відрізку [xi − 1,xi] функція S(x) є поліном третього ступеня Si(x), коефіцієнти якого треба визначити. Запишемо для зручності Si(x) у вигляді:



Тоді



Умови безперервності всіх похідних до другого порядку включно записуються як



а умови інтерполяції у вигляді



Звідси отримуємо формули для обчислення коефіцієнтів сплайну:



Якщо врахувати, що c0 = cn = 0, обчислення c можна провести за допомогою методу прогонки для тридіагональної матриці.