Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____________ д.т.н., проф. С. І. Шматков
«___» ________ 2021 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «**Імовірнісні методи побудови пробних вибірок по заданому закону розподілу**»

Виконав:

студент 4 курсу, групи КУ– 41
Галузь знань: 15 – Автоматизація та приладобудування
за спеціальністю 151 – Автоматизація та комп’ютерно-інтегровані технології.
Братусь Максим Андрійович_____

Керівник:

д.т.н., професор кафедри теоретичної та прикладної системотехніки
Угрюмов Михайло Леонідович

____

Рецензент:
д.т.н., професор кафедри моделювання систем і технологій
Ткачук Микола В’ячеславович

____

Захищено на засіданні
Атестаційної комісії № 39
протокол № __ від __.06.2021 р.
Оцінка _____ / _______________
Голова Атестаційної комісії
д.т.н., професор
_________________МІНУХІН
Сергій Володимирович

Харків – 2021

АНОТАЦІЯ

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та чотирьох додатків. Загальний обсяг роботи становить 69 сторінок, з яких 45 сторінки основної частини з 17 рисунками, 20 найменуваннями списку використаних джерел на 2 сторінках і 4 додатками на сторінках.

Об'єкт дослідження. Побудова пробних вибірок по заданому закону розподілу.

Предмет дослідження. Імовірнісні методи побудови пробних вибірок.

Ціль. Дослідження, аналіз та порівняння імовірнісних методів побудови пробних вибірок за допомогою усіченого нормального закону розподілу.

Отримані результати. Розроблено програмний продукт для автоматичної генерації багатовимірної вибірки методом латинського гіперкуба, зі збереженням параметрів розподілу в задачах прийняття рішень.

Ключові слова: генерація, метод вибірки латинського гіперкуба, класифікація, багатовимірні вибірки, імовірнісні методи.

ABSTRACT

The diploma of the robot is stored in the entry, three sections, conclusion, the list of vicarious sources and three applications. The main part of the robot is to become 69 sides, 45 sideways of the main part with 17 figures, 20 nominated listings on 2 sides and 4 additional pages on sides.

Object of study. Construction of trial samples according to a given distribution law.

Subject of study. Probabilistic methods for constructing trial samples.

Goal. Research, analysis and comparison of probabilistic methods of constructing test samples using the law of distribution.

The results obtained. A software product has been developed for automatic generation of multidimensional sampling by the Latin hypercube method, while preserving distribution parameters in decision-making problems.

Keywords: generation, Latin hypercube sampling method, classification, multidimensional samples, probabilistic methods.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

УНР – Усічений нормальний розподіл.

НР – Нормальний розподіл.

ПП – Програмний продукт.

ІМ – Імовірнісні методи.

ДМ – Детерміновані методи.

RR – Rational Rose.

КНР – Канонічний нормальний розподіл

Вступ

Сучасний етап розвитку інформаційних технологій характеризується великою кількістю накопичених даних, що являють собою певну сукупність. В залежності від обсягу сукупності і цілей дослідження можуть бути використані методи суцільного або вибіркового обстеження. При проведенні певного обстеження вивчаються всі одиниці сукупності. Такий метод може бути використаний, якщо кількість елементів сукупності невелика.

Однак, на практиці дуже часто не представляється можливим або доцільним проведення суцільного дослідження. Тому, виникає задача генерування вибірки меншого обсягу, яка дозволяє отримати інформацію про всю сукупність. Точність, з якою вибірка відображає сукупність в цілому, залежить від структури і розміру вибірки.

Розрізняють два підходи до структури вибірки – імовірнісний та детермінований.

Детермінований підхід припускає, що вибір елементів сукупності проводиться методами, заснованими методами або на міркуваннях зручності, або на рішенні дослідника.

Імовірнісний підхід передбачає, що будь-який елемент сукупності може бути обраний з певною ймовірністю. Імовірнісна вибірка більш точна ніж детермінована і дозволяє досліднику оцінити ступінь достовірності зібраних ним даних.

Тому генерація багатовимірних вибірок меншого обсягу певними імовірнісними методами є актуальною задачею і має важливе науково-практичне значення, а саме метод Латинського гіперкубу.

Об’єкт дослідження. Побудова пробних вибірок по заданому закону розподілу.

Предмет дослідження. Імовірнісні методи побудови пробних вибірок.

Ціль. Дослідження, аналіз та порівняння імовірнісних методів побудови пробних вибірок за допомогою закону розподілу.

Задачі дипломної роботи:

Аналітично усічений нормальний розподіл з використання методу Латинського Гіперкуба, згенерувати багатовимірну вибірку репрезентативна заданому закону розподілу. Провести аналіз кількості прецедентів, достатніх, щоб дана вибірка була репрезентована.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ, ЗГІДНО ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1)Імовірнісні методи

Імовірнісний метод це метод неконструктивного доведення, що, в першу чергу, використовується у комбінаториці та винайдений Паулем Ердьошем, для доведення існування наперед визначеного виду математичних об'єктів. Імовірнісні методи включають до свого складу простий випадковий відбір, систематичний, кластерний і стратифікований відбори.

Проста випадкова вибірка. Проста випадкова вибірка – це імовірнісний метод вибірки, згідно з яким кожна одиниця сукупності має відому та однакову ймовірність відбору. Тобто, всі елементи сукупності мають однаковий шанс потрапити у вибірку. Для формування даної вибірки використовують методи випадкових чисел або наосліп.

При простій випадковій вибірці з використанням статистичних таблиць випадкових значень дослідник повинен мати пронумерований список сукупності. Потім він генерує випадкові числа, щоб визначити номера одиниць, які будуть залучені у вибірці. Генерація довільних чисел відбувається випадково. В таблиці вибирають будь-яку початкову цифру і від неї рухаються в будь-якому напрямку, змінюючи напрямок руху, поки не буде обрано необхідну кількість респондентів або цю роботу виконує програма.

Перевагою методу вважається об'єктивність, тому що у всіх елементів є рівний шанс потрапити у вибірку, простота розуміння і виконання.

Недолік методу є важкість визначити кожну одиницю контуру вибірки, особливо при великих сукупностях та високі витрати на проведення.

Систематична вибірка. Систематична вибірка – це імовірнісний метод вибірки, що передбачає певну систему, відповідно до якої спочатку задається довільна початкова точка, а потім з контуру вибірки послідовно, через рівні інтервали вибираються інші елементи. В даному випадку використовують «інтервал стрибка», розрахований як співвідношення розміру контуру вибірки до обсягу вибірки.

Переваги методу – економить час і витрати; якщо генеральна сукупність володіє інформацією про досліджувану характеристику, то вибірка більш репрезентативна, ніж проста випадкова.

Недолік методу – важко визначити кожну одиницю контуру вибірки, особливо при великих генеральних сукупностях; високі витрати на проведення.

Кластерна вибірка. Кластерна вибірка заснована на розподілі сукупності на групи (кластери), кожна з яких представляє сукупність в цілому. Елементи кластера повинні бути максимально різнорідні, а самі кластери – якомога більш однорідні. В ідеалі кожен кластер повинен являти собою невелику модель генеральної сукупності. Після визначення кластерів, які будуть підлягати обстеженню, випадковим чином з елементів кластера формується вибірка. При використанні даної вибірки контур вибірки потрібен тільки для кластерів, які увійшли у вибірку. Отримані результати від кластера поширюються на всю сукупність.

Перевагою методу є – легкість застосування, тому що не потрібен список всієї сукупності; ефективність з точки зору витрат.

Недоліком є – низька точність, тому що складно сформувати неоднорідні кластери.

1.2) Детерміновані методи

Детерміновані методи відбору включають в свій склад: відбір на основі принципу зручності(нерепрезентативна вибірка), відбір на основі суджень експертів(поверхнева вибірка), формування вибірки в процесі обстеження(вибірка за принципом «снігової кулі»), а також квотний(нормований) відбір.

Відбір на основі принципу зручності. Метод, заснований на принципі зручності, полягає в тому, що дослідник формує вибірку найзручнішим для нього способом (з точки зору витрат і часу).

Переваги методу – економічність (з точки зору часу і витрат), зручність відбору.

Недолік – необ'єктивність відбору, нерепрезентативність.

Відбір на основі суджень експертів. Формування вибірки на основі суджень засновано на використанні думки компетентного фахівця (дослідника). Вважається, що дослідник, володіючи досвідом і знаннями, може досить точно відібрати одиниці вибірки, які представлятимуть генеральну сукупність.

Переваги методу – низька вартість, зручність, швидкість.

Недолік – суб'єктивність, результати залежать від компетентності дослідника.

Формування вибірки в процесі обстеження. Формування вибірки в ході обстеження полягає у визначенні первинних елементів, кожен з яких вказує на кілька нових, і так далі. Така вибірка використовується при обстеженні об'єктів зі специфічними ознаками, що займають невисоку частку в загальній множині аналогічних об'єктів, і тісно взаємодіючих між собою.

Головна завдача такої вибірки – дати оцінку незвичайним для сукупності характеристикам.

Переваги методу – дозволяє оцінити незвичайні для сукупності характеристики; невисокі витрати.

Недолік – вимагає багато часу.

Квотний(нормований) відбір. Квотна вибірка – метод детермінованої вибірки, структура якої будується за аналогією з розподілом досліджуваної ознаки в генеральній сукупності. Проводиться в два етапи. Спочатку формуються квоти з елементів генеральної сукупності відповідно до будь-якої ознаки. Потім методом зручності або на основі думок дослідника відбираються респонденти з кожної квоти. Причому квоти встановлюються таким чином, щоб частка респондентів у вибірці, що володіє певними характеристиками, відповідала частці елементів генеральної сукупності, що володіє такою ж характеристикою.

Основне допущення цього методу полягає в тому, що якщо вибірка репрезентативна за ознакою квотування, то вона буде репрезентативна і щодо розподілу думок з предмету дослідження.

Переваги методу – низька вартість; зручність вибору елементів для кожної квоти; можливість регулювати вибірку з певними характеристиками;

Недолік – велика ймовірність необ'єктивності при відборі.

Висновки до розділу 1

В даному розділі розглянуто класифікацію методів побудови пробних вибірок. Їх поділяють на детерміновані та імовірнісні. Детерміновані в свою чергу складаються з відбору на основі принципу зручності, відбору на основі суджень експертів, формування вибірки в процесі обстеження, а також квотний(нормований) відбір. Імовірнісні поділяють на просту випадкову вибірку, систематичну, кластерну та стратифіковані вибірки. На підставі розглянутої класифікації можна зробити висновок, що існує велика кількість методів для задачі побудови пробних вибірок. Однак основною метою цих методів є вибір елементів із сукупності таким чином, щоб розподіл цих елементів у вибірці якомога точніш повторював їх розподіл в сукупності, тобто вибірка була максимально репрезентативною. Досягненню цієї мети служать імовірнісні методи генерації вибірок, які й розглянуті в дипломній роботі.

РОЗДІЛ 2

ІМОВІРНІСНІ МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ

2.1)Метод статичних випробувань (Монте-Карло)

Під методом Монте-Карло розуміється чисельний метод рішення математичних задач за допомогою моделювання випадкових величин.

Метод має дві основних особливості. Перша - проста структура обчислювального алгоритму. Друга - помилка обчислень, як правило, пропорційна$\sqrt{\frac{D\xi}{N}}$, де $D\xi$ - деяка константа, а N - число випробувань.

Метод Монте-Карло особливо ефективний при вирішенні тих завдань, в яких результат потрібен з низькою точністю. Однак одну і ту ж задачу можна вирішувати різними варіантами методу Монте-Карло, яким відповідають різні значення. У багатьох задачах вдається значно збільшити точність, вибравши спосіб розрахунку, якому відповідав би значно менше значення .

Ідея Методу Монте-Карло надзвичайно проста і полягає в наступному. Замість опису випадкового явища за допомогою аналітичних залежностей, проводиться «розіграш» - моделювання випадкового явища за допомогою процедури, дає випадковий результат.

В результаті розіграшу отримуємо одну реалізацію випадкового явища. Провівши такий розіграш багато разів, отримуємо статистичний матеріал, безліч реалізацій випадкового явища, який можна обробити звичайними методами математичної статистики. По суті методом розіграшу може бути вирішена будь-яка імовірнісне завдання, однак, виправданим він стає тільки тоді коли процедура розіграшу простіше аналітичних або обчислювальних методів.

Припустимо, що нам потрібно обчислити якусь невідому величину m. Спробуємо придумати таку випадкову величину ξ, щоб Mξ = m. Нехай при цьому Dξ=b2.

Розглянемо N незалежних випадкових величин $\xi^1, \xi^2 \ldots \xi^N$(реалізацій), розподілу яких збігаються з розподілом ξ. Якщо N досить велике, то згідно центральної граничній теоремі розподіл суми $p_N = \sum_i \xi_i$ буде приблизно нормальним з параметрами $M_{p_N} = N_m, D_{p_N} = Nb^2$.

На основі Центральної граничної теореми (граничної теореми Муавра-Лапласа) можна отримати співвідношення:

$$P\left(\left|\frac{p_N}{N} - m\right| \leq k\frac{b}{\sqrt{N}}\right) = P\left(\left|\frac{1}{N}\sum_i \xi_i - m\right| \leq k\frac{b}{\sqrt{N}}\right) \to 2\Phi(k) - 1,$$

Де Фх - функція розподілу стандартного нормального розподілу.

Це надзвичайно важливе для методу Монте-Карло співвідношення. Воно дає і метод розрахунку m, і оцінку похибки.

Справді, знайдемо N значень випадкової величини ξ. Із зазначеного співвідношення видно, що середнє арифметичне цих значень буде приблизно дорівнювати m. З ймовірністю близькою до (2Ф (k) - 1) помилка такого наближення не перевищує величини $\frac{kb}{\sqrt{N}}$. Очевидно, ця помилка прагне до нуля із зростанням N.

Залежно від цілей останнє співвідношення використовується по різному:

1. Якщо взяти k=3, то отримаємо так зване «правило 3σ»:

$$P(|\frac{pN}{N} - m| \leq 3\frac{b}{\sqrt{N}}) \approx 0.9973$$

Якщо потрібен конкретний рівень надійності обчислень ,

$$P(|\frac{pN}{N} - m| \leq (\Phi^{-1}(\frac{1+\alpha}{2}))\frac{b}{\sqrt{N}}) \approx \alpha$$

Як видно з наведених вище співвідношень, точність обчислень залежить від параметра N і величини b - середньоквадратичного відхилення випадкової величини ξ.

У цьому пункті потрібно вказати на важливість саме другого параметра – b. Найкраще це показати на прикладі. Розглянемо обчислення визначеного інтеграла.

Обчислення визначеного інтеграла еквівалентно обчисленню площ, що дає інтуїтивно зрозумілий алгоритм обчислення інтеграла. Замість рівномірно розподіленої випадкової величини в цьому методі можна використовувати практично будь-яку випадкову величину, задану на тому ж інтервалі.

Отже, потрібно обчислити визначений інтеграл:

$$I = \int_a^b g(x)dx$$

Виберемо довільну випадкову величину ξ з густиною розподілу $p_{\xi}(x)$, визначеною на інтервалі (a,b). І розглянемо випадкову величину $\xi = \frac{g(\xi)}{p_{\xi}(\xi)}$.

Математичне очікування останньої випадкової величини дорівнює:

$$M(\xi) = \int_a^b \left[\frac{g(x)}{p_{\xi}(\mathrm{x})}\right] p_{\xi}(x)dx = I$$

Таким чином, отримуємо:

$$P(\left|\frac{1}{N}\sum_i \xi_i - I\right| \leq 3\sqrt{\frac{D\xi}{N}}) \approx 0.9973$$

Останнє співвідношення означає, що якщо вибрати N значень $\xi^1, \xi^2 \ldots \xi^N$, то при досить великому N:

$$\frac{1}{N}\sum_i \frac{g(\xi_i)}{p_{\xi_i}(\xi_i)} \approx I$$

Таким чином, для обчислення інтеграла, можна використовувати практично будь-яку випадкову величину ξ. Але дисперсія D ζ, а разом з нею і оцінка точності, залежить від того яку випадкову величину ξ взяти для проведення розрахунків.

Можна показати, що D ζ матиме мінімальне значення, коли $p_\xi(x)$ пропорційно |g(x)|. Вибрати таке значення $p_\xi(x)$ в загальному випадку досить складно (складність еквівалентна складності розв'язуваної задачі), але керуватися цим міркуванням варто, тобто вибирати розподіл ймовірностей за формою схожою з модулем інтегрованої функції.

Розглянемо числовий приклад: a=0; b=π/2; g(x)=cos cos x .

Обчислимо значення інтеграла з застосуванням двох різних випадкових величин.

У першому випадку будемо використовувати рівномірно розподілену випадкову величину на [a,b], тобто $P_\xi(x) = \frac{2}{\pi}$.

У другому випадку візьмемо випадкову величину з лінійною густиною на [a,b], тобто:

$$p_\xi(x) = \frac{4}{\pi}(1 - \frac{2x}{\pi})$$

Неважко бачити, що лінійна густина краще відповідає функції g(x).

На рисунку 2.1 показані графіки, зазначених функцій.

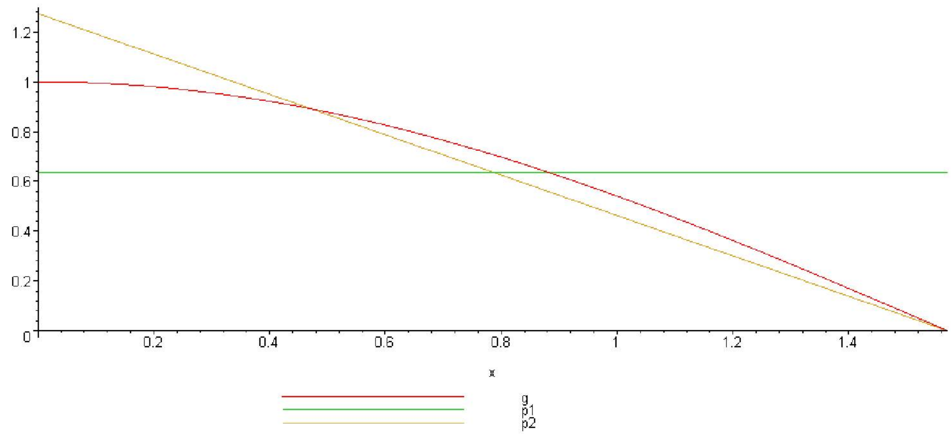


Рисунок 2.1. Графіки зазначених функцій

Точне значення інтеграла легко обчислити аналітично, воно дорівнює 1.

Результати одного моделювання при N = 10:

Для рівномірно розподіленої випадкової величини: I≈1.21666.

Для випадкової величини з лінійною густиною розподілу: I ≈0.97641.

У першому випадку відносна похибка більше 21%, а в другому 2.35%. Точність $3\sqrt{\frac{D\xi}{N}}$ в першому випадку дорівнює 0.459, а в другому - 0.123.

Цей модельний приклад показує важливість вибору випадкової величини в методі Монте-Карло. Вибравши, правильну випадкову величину, можна отримати більш високу точність обчислень, при меншій кількості ітерацій.

2.2)Метод латинського гіперкуба

Вибірка латинського гіперкуба - це статистичний метод генерації майже випадкової вибірки значень параметрів з багатовимірного розподілу. Метод вибірки часто використовується для побудови комп'ютерних експериментів або для інтеграції Монте-Карло.

При вибірці функції змінних діапазон кожної змінної ділиться на рівноімовірні інтервали. потім розміщуються точки вибірки, щоб задовольнити вимогам латинського гіперкуба; це змушує кількість поділів, бути рівним для кожної змінної. Ця схема вибірки не вимагає більшої кількості вибірок для більшої кількості вимірів (змінних); ця незалежність - одне з головних переваг даної схеми вибірки. Ще одна перевага полягає в тому, що випадкові проби можна брати по одній, запам'ятовуючи, які проби були взяті на даний момент.

У двох розмірах, відмінність між випадковою вибіркою, латинської вибіркою гіперкуби і ортогональної вибіркою може бути пояснено таким чином:

- У **випадкових** нових типових пунктах зроблені, не беручи до уваги раніше вироблені типові пункти. Потрібно не обов'язково знати заздалегідь, скільки типових пунктів необхідно.
- У **латинському гіперкуби**, потрібно спочатку вирішити, скільки зразок вказує на використання, і для кожного типового пункту пам'ятають, в якому ряді і колонці був узятий типовий пункт.
- У **ортогональній вибірці** типовий простір поділений на однакові ймовірні підмести. Всі типові пункти тоді обрані, одночасно переконавшись, що повний ансамбль типових пунктів - латинський зразок гіперкуби і що кожний підпростір вибрано з тією ж самою щільністю.

Таким чином **ортогональна вибірка** гарантує, що ансамбль випадкових чисел - дуже хороший представник реальної мінливості, **латинський гіперкуб** гарантує, що ансамбль випадкових чисел представлений для реальної мінливості, тоді як **традиційна випадкова вибірка** є просто ансамблем випадкових чисел без будь-яких гарантій.

В даному випадку всі області простору вибірки представлені вхідними значеннями і проводиться генерація вибірки розміру N із K змінних $x = [x_1 \ldots x_k]^T$ з функцією густини розподілу ймовірності $f(x)$. Розподіл густини ймовірностей, випадкової величини, що цікавить, спочатку ділиться на N діапазонів з рівною ймовірністю $\frac{1}{N}$, а величина з вибірки кожного рівного діапазону ймовірності береться тільки один раз.

Для відбору існуючих допоміжних даних ми не можемо безпосередньо застосувати МЛГ(Метод Латинського Гіперкуба) до багатовимірного розподілу. Зразки відбираються звичайними МЛГ представляють собою комбінацію багатоваріантних варіантів, які можуть не існувати в реальному світі. На рисунку 2.2 показано ділянку розсіювання кумулятивної вірогідності даних висоти та схилу у вигляді сірих точок. Ми використовували алгоритм МЛГ, щоб зробити 10 зразків латинських гіперкубів. Отримані 10 зразків відображаються як кола на рис. 2.2, ми бачимо, що лише 4 з 10 зразки збігаються з точками даних, а інші 6 зразків не існують у реальному світі. Ця проблема буде більш виражена при роботі з великою кількістю змінних та зразкових сайтів.

Це стає проблемою оптимізації: дано N публікацій з допоміжними даними (X), виділяємо n зразкових публікацій 5N, так що вибіркові с публікацій x утворюють латинський гіперкуб, або багатоваріантний розподіл X максимально стратифіковано. Це вимагає процедури пошуку, в даній роботі ми пропонуємо МЛГ з пошуковим алгоритмом на основі евристичних правил у поєднанні з графіком відпалу. Цей алгоритм працює як на безперервних, так і на категоричних даних.

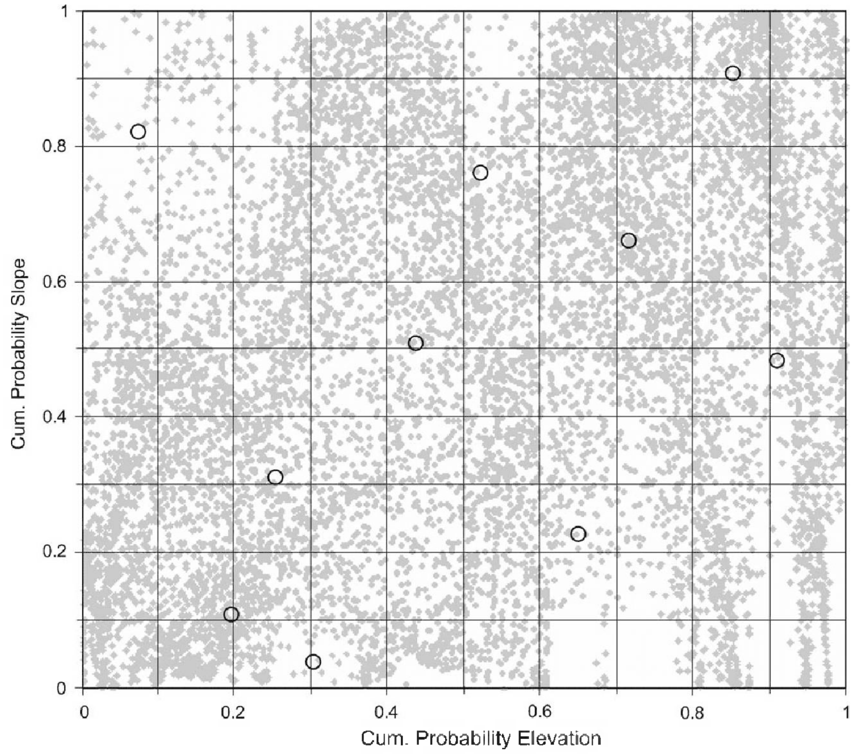


Рисунок - 2.2 Сірі точки представляють дані в області, а чорні кола представляють зразки, зроблені з використанням звичайних МЛГ.

Поки значення цільової функції не виходить за межі заданого критерію зупинки або для певного числа ітерацій.

Остаточний зразок буде представляти собою вірний або приблизний латинський гіперкуб функціонального простору, за якого збережеться розподіл та багатовимірна кореляція.

2.3)Метод генерації рідких сіток

Метод рідких сіток заснований на розкладанні d -вимірного інтеграла I на нескінченну

суму доданків і на скороченні цієї суми, яка врівноважує точність. Існують різні способи скорочення суми, відповідні різним конструкціям рідких сіток, таким як, класична конструкція, конструкція запізнювання основних послідовностей, узагальнена конструкція та розмірно-адаптивна конструкція.

Класична конструкція методу генерації рідких сіток

Розріджені сітки можуть бути визначені для загального тензорного добутку області $\Omega^d \in R^d$. Розглянемо випадок, коли $\Omega = [0,1]$, але більшість результатів можна легко

узагальнити на інші області. Зупинимо увагу на випадку, коли $\Omega = \mathrm{R}$.

Для одномірної функції $f\colon [0,1] \to R$ і послідовністі неспадаючих цілих чисел $m^k, k \in N$, нехай:

$$U_{m_k} f = \sum_{i=1}^{m_k} w_{i,k} * f(x_{k,i}),$$

позначимо послідовність одновимірних квадратурних правил з m_k точками $x_{i,k}$ й вагою $w_{i,k}$, які точково сходяться до I(f) для $k \to f$. Припустимо, що $m_1 = 1$ и $U_{m1} f = f\left(\frac{1}{2}\right)$ і визначають різницеві квадратурні формули:

$$\Delta k = U_{m_k} - U_{m_{k-1}}, U_{m0} = 0$$

для $k \geq 1$.

Нехай тепер $f\colon [0,1]^d \to R$ багатовимірна функція. Тоді d -вимірний інтеграл I(f) може бути представлений нескінченною сумою доданків:

$$I(f) = \sum_{k \in N^d} \Delta k f,$$

яка включає добутки кожної можливої комбінації одновимірних різницевих формул. Тут $k \in N^d$ позначимо, як мультиіндекс с $k_j > 0$ та

$$\Delta kf = (\Delta k_1 \dots \Delta k_d)f$$

Для даного рівня $I \in N$ класичний метод рідких сіток, часто також позначається як метод Смоляка, визначається формулою:

$$SG_l f = \sum_{|k|_1 \leq l+d-1} \Delta kf$$

$$|k|_1 = \sum_{j=1}^{d} k_j$$

З безлічі всіх можливих індексів $k \in N^d$, враховуються тільки ті, для яких ||1-norm норма менше постійної. Відзначимо, що підхід до добутку відновлюється, якщо замість ||1-norm норми, використовувати для вибору індексів норму $||_\infty = \max\{k_1 \dots k_d\}$. Запізнювання основних (базисних) послідовностей. Розглядається оптимізація класичної конструкції методу рідкої сітки щодо її поліноміальної

ступені точності.

Для цього позначимо через ϑ_l^d простір всіх багатовимірних многочленів розмірності d, які мають максимальну ступінь l. Припустимо, що одномірні формули U_{m_l} точні для простору ϑ_l^d. Тоді класичне правило квадратичних рідких сіток SGlточне для некласичного простору многочленів

$$\vartheta_l^d = \{\vartheta_{k1}^1 \dots \vartheta_{kd}^1 : |k|_1 = l + d - 1\},$$

Метод також зберігає класичну точність полінома (на основі простору ϑ_l^d) одновимірних квадратурних правил до деякої міри.

Узагальнений метод рідких сіток.

Метод рідких сіток може бути адаптований до певних класів підінтегральних функцій, якщо доступна апріорна інформація про важливість розмірів або важливість взаємодії між вимірами.

Це досягається шляхом вибору відповідних наборів індексів $\vartheta \in d$. Для забезпечення достовірності розширення суми доданків, безліч індексів ϑ повинні задовольняти умові допустимості:

$$k \in \vartheta \text{ и } l \leq k \Rightarrow l \in \vartheta,$$

де $l \leq k$ визначається формулою $l_j \leq k_j$ для j=1,…,d.

Таким чином, узагальнений метод розріджених сіток виглядає наступним чином:

$$SG_{\vartheta}f = \sum_{k \in \vartheta} \Delta k f.$$

2.4)Усічений нормальний розподіл

Нормальний розподіл випадкової величини має необмежену область розсіювання. Так само область значень випадкової величин, аналіз котрих приходиться проводить в природознавстві, техніці та економіці, виявляється кінцевою. Як слідство, розподілу даних випадкових величин відрізняються від ідеалізованої моделі з необмеженою областю розсіювання.

Найбільш очевидною фізичною моделлю випадкового процесу з обмеженою областю розсіювання служать пісочні часи, в яких пісок із «точкового» джерела випадає на горизонтальну площину, обмежену непроникними, абсолютно пружними вертикальними стінками. При достатньому видаленні стінок від джерела вони не будуть впливати на формування купи піску, тому її форма буде симетричною. В протилежному випадку піщинки будуть відбиватися від стінок, що призведе до відмінності форми купи від нормальної.

Як приклади фізичних характеристик з обмеженою областю розсіювання можна привести щільність вугілля, що змінюється в діапазоні від ρ_{min} до деякого максимального значення ρ_{max}, час безвідмовної роботи групи однотипних приладів, що змінюється в діапазоні від деякого мінімального значення T_{min} до деякого максимального T_{max} тощо.

Класична модель нормального закону.

Еквівалентність рішення рівняння Фоккера-Планка, що описує процес дифузії:

$$\frac{\partial}{\partial t} f(t,x) = D\frac{\partial^2}{\partial x^2} f(t,x),$$

Де $f(t,x)$ – щільність розподілу частинок $f(t,x) = \frac{\partial}{\partial t}\mathrm{P}(t,x)$, $\mathrm{P}(t,x)$ - вірогідність знаходження частинок в момент часу t в точці x; D – коефіцієнт дифузії для необмеженої області розсіювання броунівських частинок було знайдено А. Ейнштейном.

$$f(t,x) = \frac{n}{\sqrt{4\pi D}} * \frac{e^{\frac{x^2}{4Dt}}}{\sqrt{t}}$$

Тут n – нормований коефіцієнт, вибираємо з умови $\int_{-\infty}^{\infty} f(t,x)dx = 1$. Трактуючи величину випадкового процесу для конечного часу t, отримуємо математичну модель нормального закону розподілу:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} * e^{\frac{x^2}{2\sigma^2}}$$

Аналогічний результат можна отримати, розглянувши броунівський рух точкової частинки на нескінченній прямій під дією випадкових поштовхів, що викликають з ймовірністю ρ зміщення в одну сторону, і з імовірністю q зміщення в інший бік.

Імовірність $\mathrm{P}_n(m)$ того, що після n випадкових поштовхів частка виявиться в m точці розраховується за формулою:

$$\mathrm{P}_n(m) = \frac{n!}{\left(\frac{n+m}{2}\right)!\left(\frac{n-m}{2}\right)!} * p^{\frac{m+n}{2}} * q^{\frac{m-n}{2}}$$

Де $n + m$ $n - m$ – число поштовхів, отже елементарних зміщень(кроків) вправо і вліво, відповідно. При $p = q = \frac{1}{2}$

$$\mathrm{P}_n(m) = C_n^{\frac{n+m}{2}} * \left(\frac{1}{2}\right)^2.$$

Усічений нормальний розподіл.

Усічений нормальний розподіл традиційно використовується в задачах оцінки надійності технічних систем і точності виробництва. УНР $N'(m,s)$ має випадкову послідовність {x’} витягнута з випадкової послідовності {x}, що має нормальний розподіл $N(m,s)$, кожен елемент якої задовольняє таким умовам:

$$x_{min} \le x'_k \le x_{max},$$

Де x_{min}, x_{max} – точки усічення.

Далі представлена таблиця з основними характеристиками УНР.

Назва характеристики	Формула або числове значення характеристики
Позначення	$N'(m, s, q, x_{min}, x_{max})$
Параметри	m, s, x_{min}, x_{max}
Щільність розподілу	$j'_N(x; m, s, a_1, a_2) = \frac{1}{S} * \frac{j\left(\frac{x-m}{S}\right)}{F\left(\frac{x_{max}-m}{S}\right) - F(\frac{x_{min}-m}{S})}$ Де j(x) – щільність стандартного нормального розподілу N(0,1); $F(x) = \frac{1}{\sqrt{2p}} \int_{-\infty}^{x} e^{\frac{x^2}{2}} dx$ – функція Лапласа, m,s – параметри вихідного нормального розподілу, $x_{min} < x_{max}$ – точки усічення.
Функція розподілу	$F'_N(x; m, s, x_{min}, x_{max}) = \int_{a_1}^{x} j'(x; m, s, x_{min}, x_{max}) dx = \frac{1}{S} * \frac{1}{F\left(\frac{x_{max}-m}{S}\right) - F(\frac{x_{min}-m}{S})} \int_{a_1}^{x} j\left(\frac{x-m}{x}\right) dx = \frac{F\left(\frac{x-m}{s}\right) - F\left(\frac{a_1-m}{s}\right)}{F\left(\frac{x_{max}-m}{S}\right) - F(\frac{x_{min}-m}{S})}$
Середнє значення	$M[x] = m - (l_2 - l_1) * s,$ Де $l_1 = \frac{j(x_1)}{F(x_2)-F(x_1)}$, $l_2 = \frac{j(x_2)}{F(x_2)-F(x_1)}$, $x_1 = \frac{x_{min}-m}{s}$, $x_2 = \frac{x_{max}-m}{s}$
Дисперсія	$D[x] = (1 + l_1 x_1 - l_2 x_2 + (l_1 + l_2)^2) * s^2$

З таблиці видно, що УНР $N'(m, s, q, x_{min}, x_{max})$ також відноситься до класу чотирьох параметричних розподілів.

Порівняємо неусічений нормальний розподіл з УНР.

На рис. 2.4.1 наведені залежності відносин числових характеристик усіченого і неусіченого нормального розподілів і значення нормує множники C0 від відносини mt / σt.

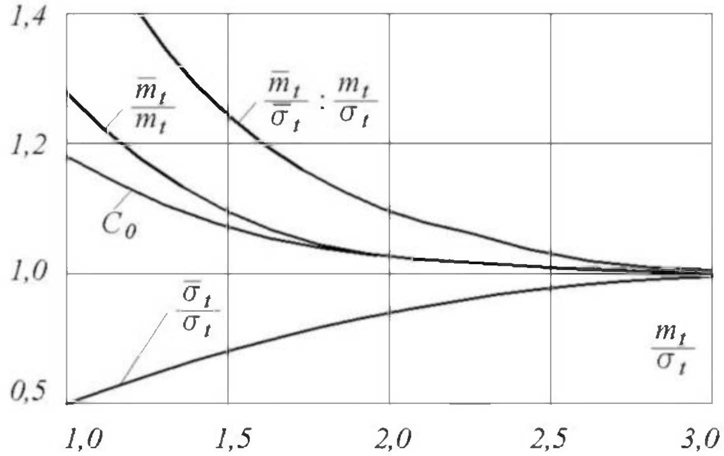


Рис. 2.4.1. Залежність відносин числових характеристик усіченого і неусіченого нормального розподілу і нормує множники C0 від відносини mt / σt. (Лінія над m_t , σt означає, що ці характеристики відносяться до усіченому розподілу)

Зважаючи на велику теоретичного і прикладного значення нормального розподілу його намагаються іноді застосувати і при явно несиметричних розподілах напрацювання до відмови. Для цього підбирають деяку функцію випадкової напрацювання до відмови, наприклад lg T, T2 тощо. Приблизно таку нормальному закону. Наприклад, досить часто використовується логарифмічно нормальний розподіл втомної довговічності, при якому передбачається, що логарифм числа циклів навантаження до руйнування зразка розподілений за нормальним законом.

Аналітичний вираз для усіченого нормального розподілу можна отримати, якщо знайти μ (математичне очікування) та σ (дисперсію) для Канонічного нормального розподілу(КНР). Для початку задаємо початкове приближення:

$$\mu^0 = \bar{\mu}\ ;\ \sigma^0 = \bar{\sigma}$$

З самого початку циклу ми робимо інкремент $n = n + 1$ для того, щоб коли ми поверталися знову до n прибавлялось 1.

Наступним кроком треба дізнатися математичне очікування неперервної випадкової величини $\mathrm{M}[x]$ та дисперсію неперервної випадкової величини $\sqrt{D[x]}$ наступним чином:

$$\bar{\mu}^{n+1} = \mathrm{M}[x] = \int_{-\infty}^{+\infty} x f(x) dx$$

$$\bar{\sigma}^{n+1} = \sqrt{D[x]} = \int_{-\infty}^{+\infty} (x - m_x)^2 f_x(x) dx$$

Наступна формула вказує що, якщо умова відповідає поставленій задачі ми робимо розрахунки для μ^{n+1} та σ^{n+1}, а якщо ні, ми повертаємося на початок розрахунку коли до нашої n прибавляється 1:

$$\frac{|\bar{\mu}^{n+1} - \bar{\mu}^n|}{\bar{\mu}^n} > \varepsilon\ or\ \frac{|\bar{\sigma}^{n+1} - \bar{\sigma}^n|}{\bar{\sigma}^n} > \varepsilon$$

Якщо дана умова відповідає нашим розрахункам ми переходимо до наступних розрахунків:

$$r^{n+1} = \alpha * r^n + 1$$

$$\mu^{n+1} = \mu^n + \frac{1}{r^{n+1}} (\bar{\mu} - \bar{\mu}^{n+1})$$

$$\sigma^{n+1} = \sigma^n + \frac{1}{r^{n+1}} (\bar{\sigma} - \bar{\sigma}^{n+1})$$

Висновки до розділу 2

В даному розділі розглянуто основні імовірнісні методи побудови пробних вибірок, такі як – метод Монте-Карло, Латинський гіперкуб, методи генерації рідких сіток та закон розподілу усічений нормальний розподіл. На підставі зробленого огляду для рішення задачі побудови пробної вибірки в задачах прийняття рішень обрано метод вибірки латинського гіперкубу за усіченим нормальним законом розподілу, який детально розглянуто в третьому розділі дипломної роботи.

РОЗДІЛ 3
РОЗРОБКА БАГАТОВИМІРНОЇ ВИБІРКИ ПО УСІЧЕНОМУ НОРМАЛЬНОМУ ЗАКОНУ

3.1)Вибір мови програмування

На даний момент існує велика кількість мов програмування, які ставлять перед розробником проблему вибору, засновану на різних факторах, таких як продуктивність та зручність.

Більшість мов програмування виникли виходячи з конкретних вимог деякої предметної області, але треба розуміти, що не існує якоїсь однієї мови програмування, яка є найкращим вибором. Можна віддати перевагу певним чинникам, таким як продуктивність і безпека програмного продукту, в порівнянні з іншими факторами, такими як, наприклад, кількість рядків коду.

Правильний вибір дозволить розробнику створити компактне, просте рішення з легкою документацією.

При виборі мови програмування враховуються такі чинники:

- цільова платформа;
- гнучкість мови;
- час виконання проекту;
- продуктивність.

На підставі проведеного порівняльного аналізу (табл. 4.1) мов програмування у якості мови використаної в дипломній роботі було обрано Microsoft Visual C#.

C# – об'єктно-орієнтована мова програмування, призначена для розробки різноманітних програм, які виконуються в середовищі .NET Framework. За допомогою мови C# можна створювати звичайні програми Windows, програми "клієнт-сервер", програми баз даних і т.д. Visual C# надає користувачеві розвинений редактор коду, конструктори зі зручним для використання інтерфейсом, вбудований відладчик і безліч інших засобів, що спрощують розробку програм на базі мови C# і .NET Framework.

Перевагами даної мови є її кросплатформенність і наявність безлічі бібліотек, які можуть бути скомпільовані на множинах платформ, виграш в швидкості роботи з великими обсягами даних.

У якості середовища розробки під платформу Windows було обрано візуальне середовище розробки Visual Studio.

3.2)Опис програмного продукту

Програмний продукт (ПП) призначений для побудови пробних вибірок даних імовірнісним методом латинського гіперкуба. Перевагами даної програми є генерація вибірок меншого обсягу зі збереженням імовірнісних характеристик. Програма надає перегляд того, яку кількість альтернативних рішень необхідно згенерувати для отримання бажаних результатів, а також графік порівняння кінцевих та початкових параметрів.

ПП розроблений в програмному середовищі Microsoft Visual Studio 2017, за допомогою мови програмування C#.

Rational Rose являє собою CASE засіб проектування і розробки інформаційних систем і програмного забезпечення для управління підприємствами. Як і інші CASE засоби (ARIS, BPwin, ERwin) його можна застосовувати для аналізу і моделювання процесів. Особлива відмінність Rational Rose від інших засобів полягає в об'єктно-орієнтованому підході. Графічні моделі, що створюються за допомогою цього засобу, засновані на об'єктно-орієнтованих принципах і мові UML (Unified Modeling Language). Інструменти моделювання Rational Rose дозволяють розробникам створювати цілісну архітектуру процесів підприємства, зберігаючи всі взаємозв'язки та управляючі між різними рівнями ієрархії.

UML – мова для визначення, уявлення, проектування та документування програмних систем, організаційно-економічних систем та інших систем різної природи.

Призначення UML: візуалізація, конструювання, специфікування, документування.

У стандарті UML присутній наступний набір діаграм: діаграми варіантів використання (use case diagrams), діаграма класів (class diagrams), (behavior diagrams), діаграми взаємодії (interaction diagrams) та ін.

Діаграма варіантів використання – це послідовність дій, виконуваних системою у відповідь на подію, що ініціюється деякою зовнішньою особою.

Дійова особа (actor) – це роль, яку користувач відіграє по відношенню до системи. Дійова особа – це фізична особистість, або користувач системи. Вона є найбільш типово і знаходиться майже в кожній системі.

Варіант використання описує типову взаємодію між користувачем і системою, це те, що користувач очікує від системи. У процесі аналізу предметної області було виявлено, що в системі можливий один актор. Він має повні привілеї на запуск програми і зміну початкових даних.

Існують два потоки: основний і альтернативний.

Основний – у користувача запитуються параметри, необхідні для введення в діалогове вікно. Альтернативний – не заповнено будь-яке поле, внаслідок чого на екрані з'являється повідомлення про некоректність введених даних.

Якщо дані введені правильно – програма проводить розрахунок.
Діаграма використання для даного проекту наведена на рисунку 3.2.1.

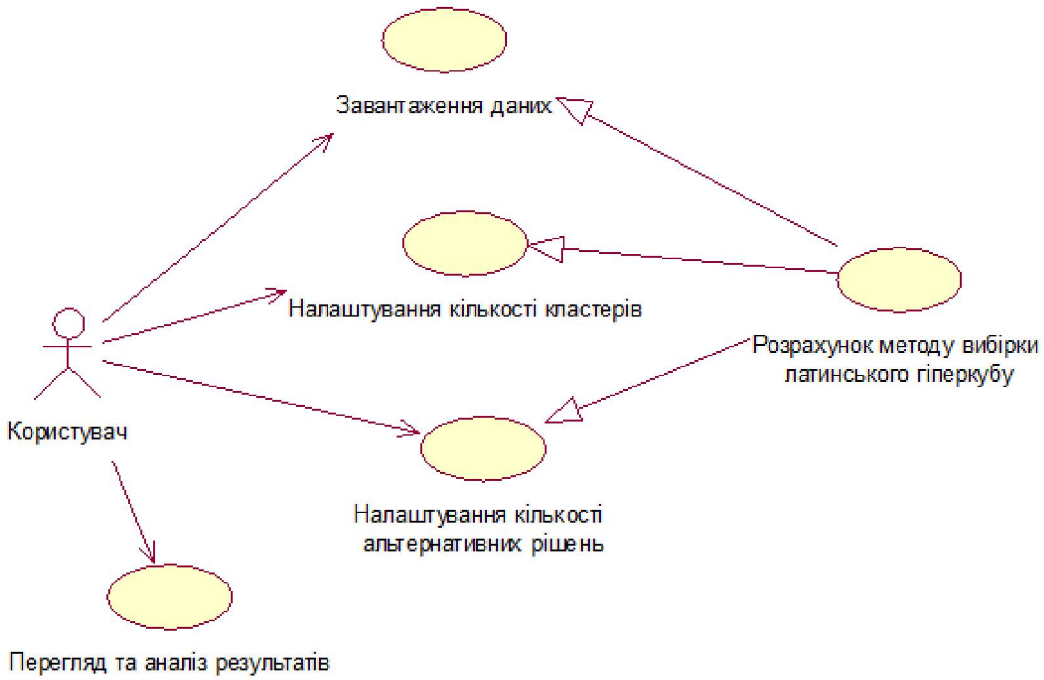


Рисунок 3.2.1 – Діаграма використання

Діаграми взаємодії описують поведінку взаємодіючих груп об'єктів. Вони охоплюють взаємодії в рамках одного варіанту використання.

Діаграми взаємодії:

– діаграми послідовності (sequence) спрямовані на тимчасове відображення взаємодії (рис. 3.2.2);

– діаграми кооперацій (collaboration) спрямовані на об'єктне відображення взаємодії (рис. 3.2.3).

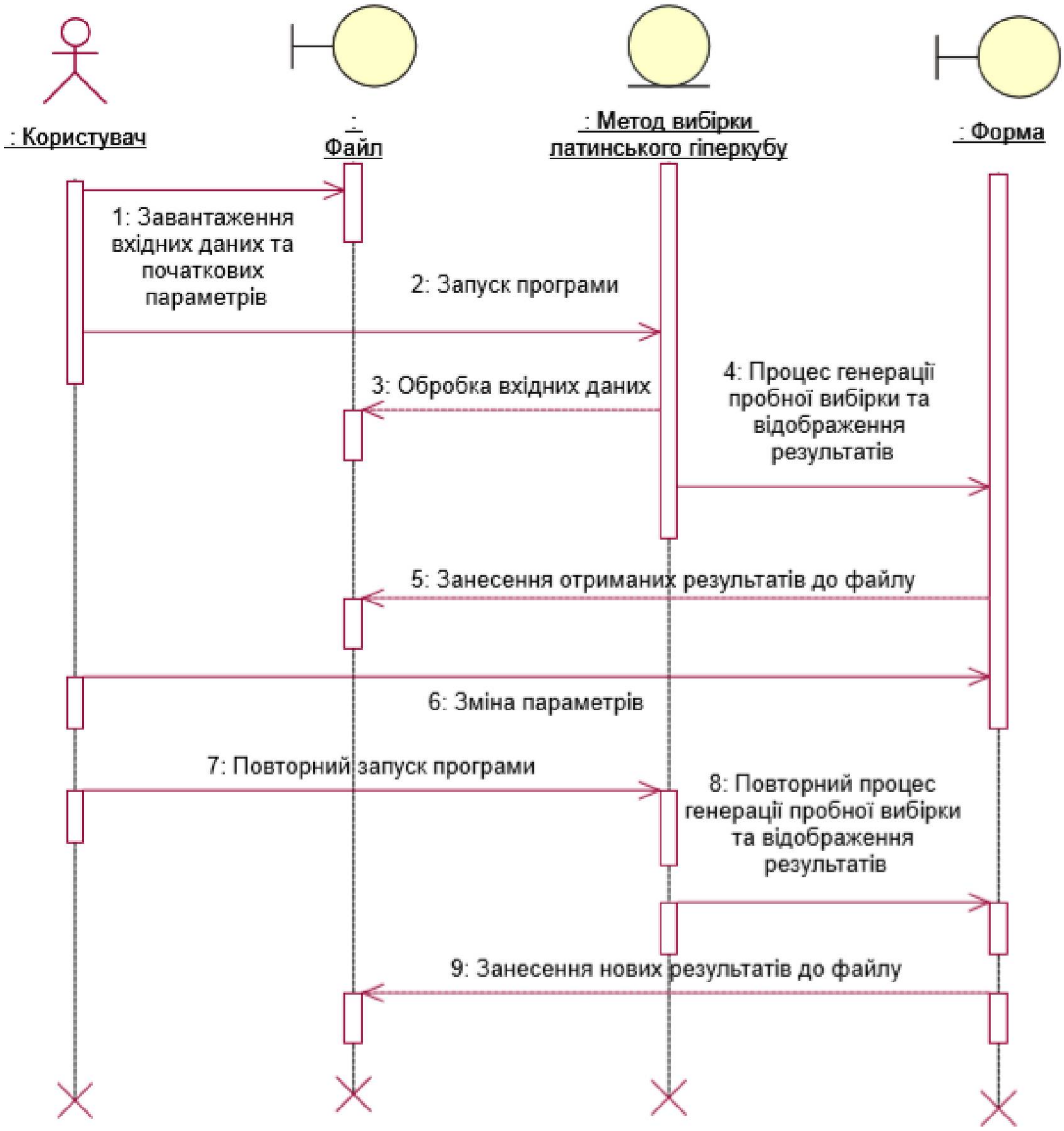


Рисунок 3.2.2 – Діаграма послідовності подій в системі

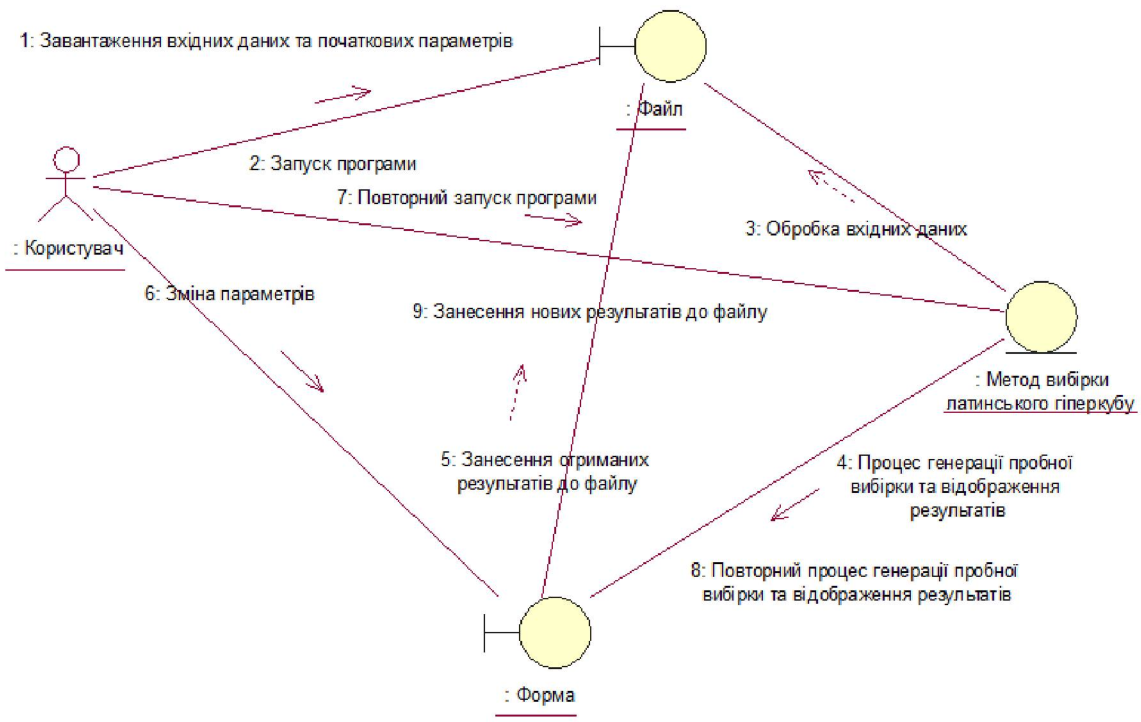


Рисунок 3.2.3 – Діаграма кооперацій

Діаграма класів визначає типи класів і статичні зв'язки, які існують між ними (рис. 3.2.4).

Класи – це типи об'єктів, які містять дані і поведінку, що впливає на ці дані.

Діаграми класів можуть розглядатися з трьох точок зору:

– концептуальна: розглядається поняття предметної області і не залежить від засобів реалізації програмного продукту.

– специфікація: розглядаються інтерфейси, а не внутрішня реалізації класів.

– реалізація: визначає реалізацію класу програмного продукту.

Стереотип – це механізм, що дозволяє розділяти класи на категорії, це спрощує навігацію по проекту та управління проектом.

Стереотипи UML:

– граничні (Boundary): класи, які знаходяться на кордоні системи і навколишнього середовища (форми, інтерфейси, звіти);

– сутності (Entity): містить інформацію яка зберігається в системі постійно (зберігають бізнес-логіку);

– керуючі (Control): відповідають за координацію дій інших класів (багато повідомлень надсилається, мало отримують).

У процесі аналізу предметної області та проектування були виявлені наступні класи:

– клас «M_Sigma» – містить поля математичного очікування та СКО вхідних параметрів.

– клас «MainForm» – забезпечує функціонування інтерфейсу користувача, завантаження вхідних даних та налаштування параметрів, а також реалізацію методу вибірки латинського гіперкубу та виведення результатів на екран.

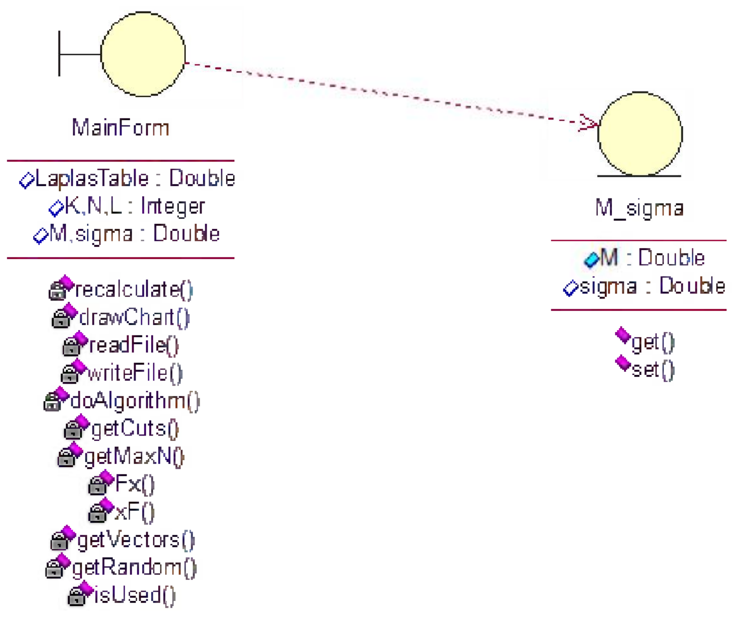


Рисунок 3.2.4 – Діаграма класів

Клас «MainForm» містить такі функції:

– readFile() – функція, яка забезпечує зчитування вхідних даних та допоміжної таблиці інтегральної функції Лапласу;

– writeFile() – функція запису отриманих результатів в файл;

doAlgorithm() – головна функція реалізації методу вибірки латинського гіперкубу.

– getCuts() – функція, яка реалізує поділ нормального розподілу заданих параметрів на необхідну кількість інтервалів;

– Fx() – допоміжна функція для визначення інтервалів;

– Xf() – допоміжна функція для визначення інтервалів;

– getVectors() – функція, яка генерує альтернативні рішення;

– getRandom() – функція, для заповнення векторів рандомними значеннями;

– isUsed() – функція для визначення того, чи були вже використані значення масиву;

– recalculate() – функція повторного розрахунку алгоритму;

– drawChart() – функція відображення отриманих результатів на формі.

3.3) **Інструкція по використанню програмного продукту**

На основі поставленої задачі в середовищі Visual Studio 2019 Community мовою C# був розроблений програмний продукт, який являє собою реалізацію методу вибірки латинського гіперкубу по закону усічений нормальний розподіл.

Перед початком роботи з програмою користувачу необхідно завантажити вхідні дані до файлу, та встановити бажані параметри для роботи (рис. 3.3.1).

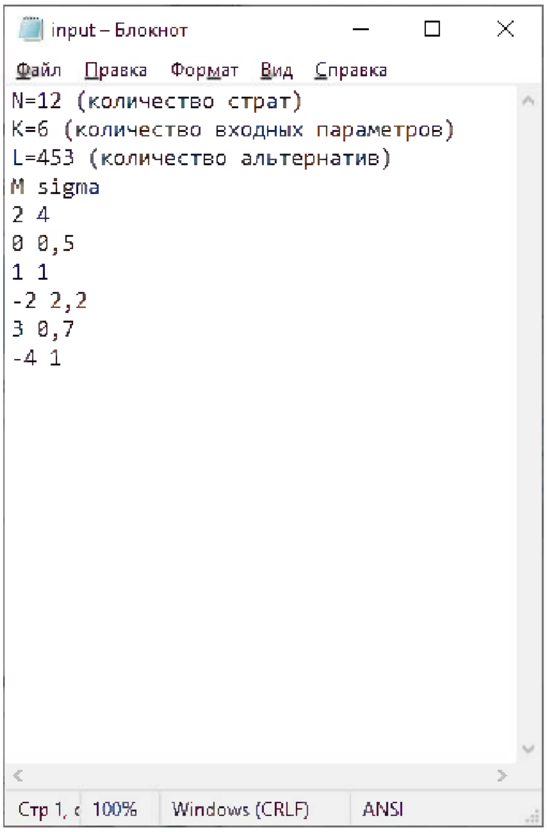


Рисунок 3.3.1 – Вхідний файл

Запустивши програму користувачем відкривається вікно форми, на якому відображається графічний результат роботи методу (рис. 3.3.2)

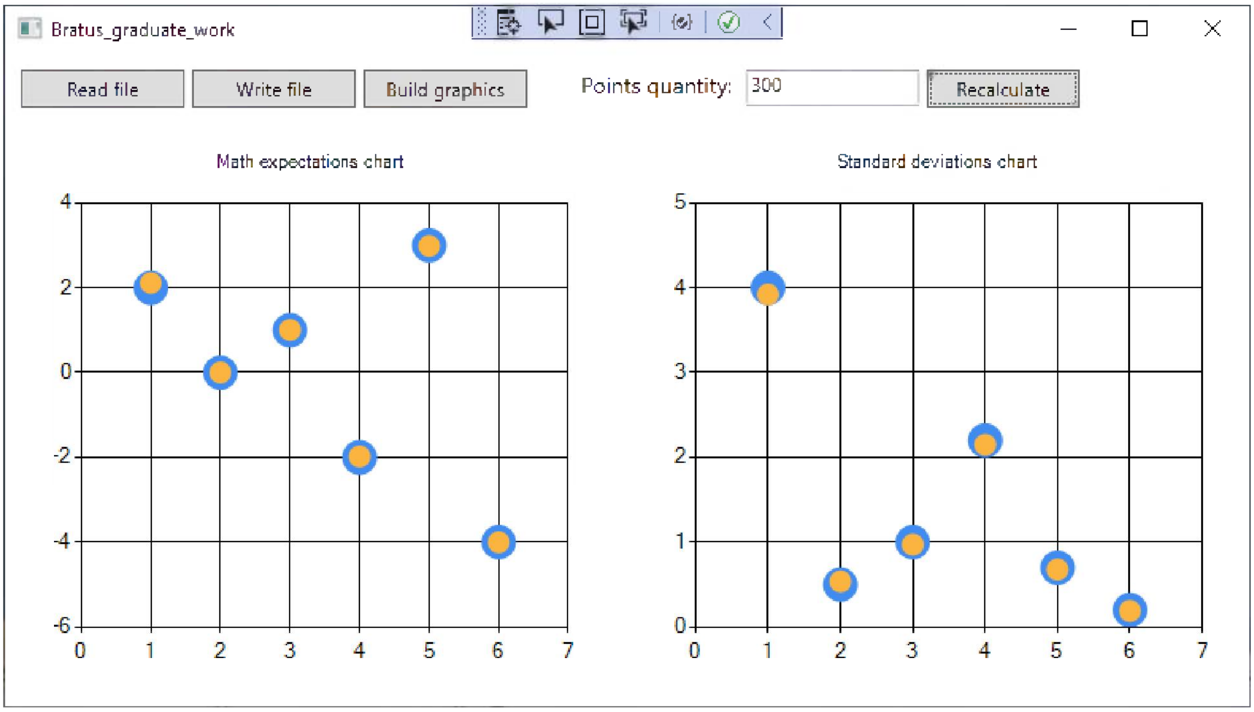


Рисунок 3.3.2 – Вікно форми

а також можна переглянути вихідний файл, до якого будуть записані отримані аналітичні результати, а саме – значення відрізків, на які поділено вхідні розподіли даних (рис. 3.3.3), альтернативні рішення для математичного очікування (рис. 3.3.4), та середньоквадратичного відхилення (рис. 3.3.5), а також результуючі значення математичного очікування та СКВ (рис. 3.3.6).

out.txt*

```
Отрезки для пары №1: (-1,79769313486232E+308, -6,24) (-6,24, -5) (-5, -4,2) (-4,2, -3,6) (-3,6, -3,12) (-3,12, -2,68) (-2,68, -2,32) (-2,32, -1,96) (-1,96, -1
Отрезки для пары №2: (-1,79769313486232E+308, -1,03) (-1,03, -0,875) (-0,875, -0,775) (-0,775, -0,7) (-0,7, -0,64) (-0,64, -0,585) (-0,585, -0,54) (-0,54, -0,
Отрезки для пары №3: (-1,79769313486232E+308, -1,06) (-1,06, -0,75) (-0,75, -0,55) (-0,55, -0,4) (-0,4, -0,28) (-0,28, -0,17) (-0,17, -0,08) (-0,08, 0,01) (0,
Отрезки для пары №4: (-1,79769313486232E+308, -6,532) (-6,532, -5,85) (-5,85, -5,41) (-5,41, -5,08) (-5,08, -4,816) (-4,816, -4,574) (-4,574, -4,376) (-4,376,
Отрезки для пары №5: (-1,79769313486232E+308, 1,558) (1,558, 1,775) (1,775, 1,915) (1,915, 2,02) (2,02, 2,104) (2,104, 2,181) (2,181, 2,244) (2,244, 2,307) (2
Отрезки для пары №6: (-1,79769313486232E+308, -4,412) (-4,412, -4,35) (-4,35, -4,31) (-4,31, -4,28) (-4,28, -4,256) (-4,256, -4,234) (-4,234, -4,216) (-4,216,
Отрезки для пары №7: (-1,79769313486232E+308, -8,42) (-8,42, -6,25) (-6,25, -4,85) (-4,85, -3,8) (-3,8, -2,96) (-2,96, -2,19) (-2,19, -1,56) (-1,56, -0,93) (-
Отрезки для пары №8: (-1,79769313486232E+308, -6,18) (-6,18, -5,25) (-5,25, -4,65) (-4,65, -4,2) (-4,2, -3,84) (-3,84, -3,51) (-3,51, -3,24) (-3,24, -2,97) (-
Отрезки для пары №9: (-1,79769313486232E+308, 3,382) (3,382, 3,475) (3,475, 3,535) (3,535, 3,58) (3,58, 3,616) (3,616, 3,649) (3,649, 3,676) (3,676, 3,703) (3
Отрезки для пары №10: (-1,79769313486232E+308, -1,824) (-1,824, -1,7) (-1,7, -1,62) (-1,62, -1,56) (-1,56, -1,512) (-1,512, -1,468) (-1,468, -1,432) (-1,432,
Отрезки для пары №11: (-1,79769313486232E+308, 7,086) (7,086, 7,675) (7,675, 8,055) (8,055, 8,34) (8,34, 8,568) (8,568, 8,777) (8,777, 8,948) (8,948, 9,119) (
Отрезки для пары №12: (-1,79769313486232E+308, -22,802) (-22,802, -20,725) (-20,725, -19,385) (-19,385, -18,38) (-18,38, -17,576) (-17,576, -16,839) (-16,839,
```

Рисунок 3.3.3 – Значення відрізків

Альт.(№)	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12
1	1,98774	1,99168	1,95063	1,99352	1,93991	1,98300	1,95362	1,95670	1,95791	1,97836	1,97749	1,96556
2	-0,00345	-0,00206	-0,00294	-0,00291	-0,00225	-0,00424	-0,00034	-0,00462	-0,00368	-0,00105	-0,00317	-0,00265
3	0,98570	0,98575	0,99555	0,99243	0,99548	0,99351	0,98577	0,99248	0,99527	0,99951	0,98795	0,98883
4	-2,02353	-2,02756	-2,03820	-1,99332	-2,00834	-2,02804	-2,03368	-2,01110	-2,00276	-2,01582	-2,00930	-2,03074
5	2,99015	2,99161	2,99166	2,99536	2,99758	2,99747	2,99659	2,99534	2,99121	2,99013	2,99749	2,99719
6	-4,00215	-4,00313	-4,00201	-4,00170	-4,00302	-4,00160	-4,00029	-4,00205	-4,00158	-4,00325	-4,00162	-4,00110
7	5,95856	5,97797	5,94044	5,91977	5,94300	5,93250	5,91799	5,91387	5,92624	5,89758	5,94191	5,94518
8	-0,01793	-0,01701	-0,01937	-0,04373	-0,01654	-0,00929	-0,02078	-0,03437	-0,01671	-0,02916	-0,04087	-0,03142
9	3,99690	3,99727	3,99672	3,99793	3,99765	3,99848	3,99890	3,99499	3,99956	3,99692	3,99739	3,99535
10	-1,00124	-1,00327	-1,00082	-1,00157	-1,00599	-1,00494	-1,00163	-1,00331	-0,99970	-1,00352	-1,00226	-1,00144
11	10,99563	10,99476	10,98422	10,98266	11,00545	10,99029	10,98147	10,98484	10,98457	10,98425	10,98216	10,97761
12	-9,07917	-9,08597	-9,05474	-9,06340	-9,03089	-9,02280	-9,02753	-9,02192	-9,04285	-9,07865	-9,01605	-9,04511

Рисунок 3.3.4 – Альтернативні рішення для математичного очікування

Альт.(№)	sigma1	sigma2	sigma3	sigma4	sigma5	sigma6	sigma7	sigma8	sigma9	sigma10	sigma11	sigma12
1	3,47795	3,43891	3,50970	3,46015	3,50493	3,43277	3,46300	3,44765	3,47016	3,50240	3,47182	3,45785
2	0,43422	0,43322	0,43448	0,42896	0,42921	0,43866	0,43340	0,43648	0,43376	0,43060	0,42820	0,43205
3	0,87297	0,85984	0,87612	0,86466	0,86651	0,87759	0,87100	0,86682	0,86215	0,87684	0,86396	0,87540
4	1,89779	1,92355	1,91699	1,92259	1,89954	1,92206	1,91492	1,91613	1,90765	1,92526	1,88919	1,92926
5	0,61075	0,60739	0,61434	0,60992	0,61683	0,61231	0,61566	0,60388	0,60399	0,60714	0,60523	0,60749
6	0,17590	0,17309	0,17584	0,17619	0,17307	0,17233	0,17232	0,17486	0,17473	0,17599	0,17404	0,17199
7	6,06175	6,09138	6,18039	6,19038	6,12872	6,09373	6,06834	6,12937	6,09401	6,09361	6,10329	6,11987
8	2,62067	2,57355	2,64661	2,60553	2,57705	2,62243	2,60877	2,62968	2,62788	2,64529	2,57496	2,60844
9	0,25870	0,26185	0,26329	0,25824	0,25932	0,25834	0,25786	0,26319	0,26248	0,26139	0,26396	0,26320
10	0,34142	0,34998	0,35178	0,35015	0,34539	0,34452	0,34843	0,34512	0,34564	0,35308	0,34441	0,34466
11	1,63467	1,65761	1,63792	1,66335	1,66528	1,65325	1,66004	1,66562	1,65780	1,65692	1,64088	1,63429
12	5,85481	5,89740	5,84059	5,86303	5,72670	5,94285	5,86383	5,82932	5,82122	5,93708	5,80782	5,84116

Рисунок 3.3.5 – Альтернативні рішення для середньоквадратичного відхилення

M	S
1,97022	3,47797
-0,00349	0,43365
0,99140	0,87136
-2,01789	1,91440
2,99495	0,61003
-4,00185	0,17453
5,93607	6,11385
-0,02232	2,61138
3,99763	0,26072
-1,00251	0,34753
10,98582	1,65434
-9,04830	5,84134

Рисунок 3.3.6 – Результуючі значення для математичного очікування та середньоквадратичного відхилення

Після виконання роботи програми користувач може повторно розрахувати вибірку латинського гіперкубу, змінивши на формі кількість відрізків, на які необхідно поділити вхідні розподіли (N), враховуючи встановлені мінімальне та максимальне значення даного параметру (рис. 3.3.7), та встановити бажану кількість альтернативних рішень (L) (рис. 3.3.7).

Рисунок 3.3.7 – Зміна параметрів

Після встановлення бажаних параметрів користувачеві необхідно натиснути кнопку «Перерозрахувати» (рис. 3.3.8), та при бажанні занести отримані результати до вихідного файлу, для чого натиснути кнопку «Записати до файлу» (рис. 3.3.8).

Рисунок 3.3.8 – Збереження та відображення даних

В програмі є кнопка построїти графік (рис. 3.3.9), яка відкриває вікно з побудованими діаграмами по графіку, які заповнені точками по усіченому закону розподілу методом латинського гіперкубу (рис. 3.3.10)

Рисунок 3.3.9 – побудувати графіки

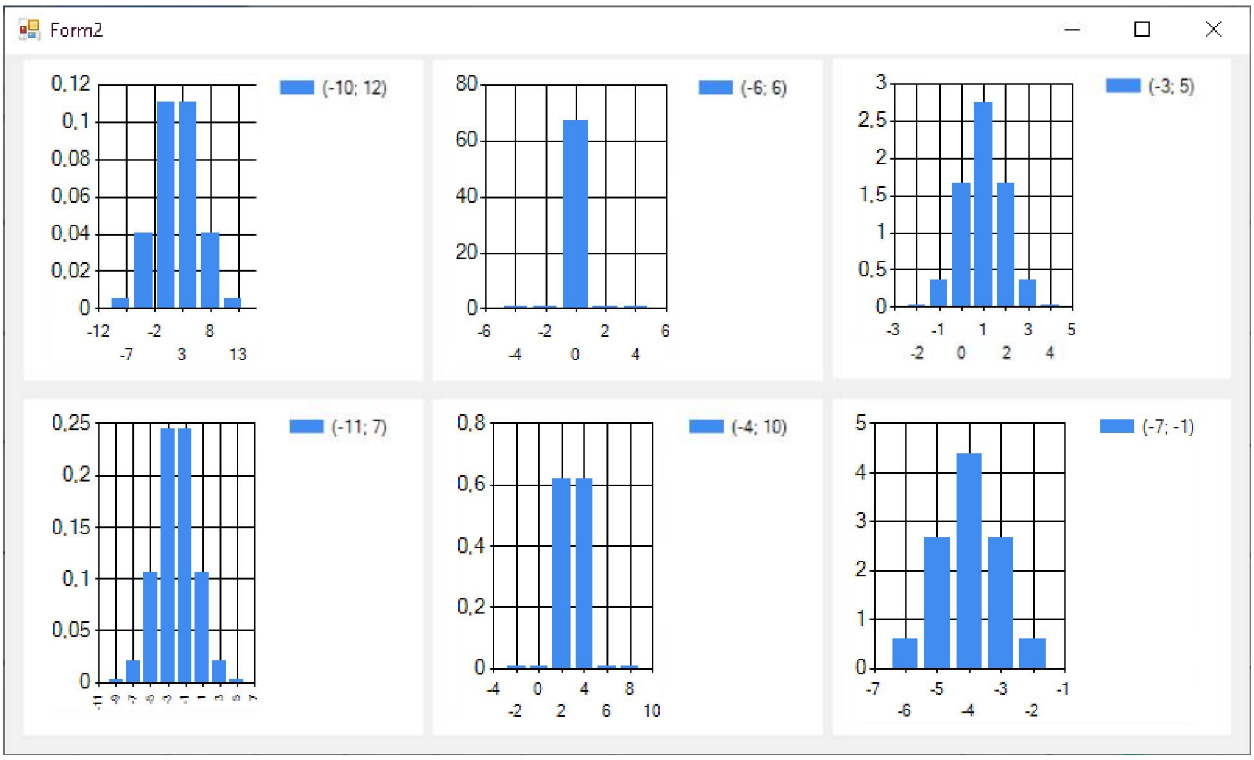


Рисунок 3.3.10 - побудованих діаграм по графіку

Висновок до 3 розділу

В даному розділі було обґрунтовано вибір мови програмування, описано проектування програмного продукту, розроблено проектування системи в середовищі RationalRose та надані такі UML діаграми: діаграма варіантів використання, діаграма взаємодії та діаграма класів. Також надана інструкція з використання розробленого програмного продукту.

ВИСНОВКИ

У кваліфікаційні роботі було проведено процес розробки вибірки методом Латинського гіперкубу. Розглянуто такі основні методи, як метод Монте-Карло, метод Латинський гіперкуб, методи генерації рідкісних сіток. Для вирішення поставленої задачі обрано метод вибірки латинського гіперкуба, представлена його математична модель та розроблено програмний продукт для даного алгоритму. Даний метод перевершує інші щодо своєї швидкої збіжності та стійкості отриманих результатів.

У першому розділі було порівняно імовірнісні та детерміновані методи, імовірнісні методи є практичніше та надійніше.

У другому розділі проведено аналіз між імовірнісними методами і був обраний для роботи метод латинського гіперкубу.

У третьому розділі було описано програмний продукт до заданого завдання, вибрана мова програмування.

Отже, задача, що полягала в побудові багатовимірної вибірки методом латинського гіперкубу за законом усіченого нормального розподілу зі збереженням характеристики розподілу вирішена в повному обсязі і мета досягнута.

Список Джерел

1) The generalization of Latin hypercube sampling / Michael Shields, Jiaxin Zhang.

2) Estimates of the coverage of parameter space by Latin Hypercube and Orthogonal sampling: connections between Populations of Models and Experimental Designs / Diane Donovan, Kevin Burrage, Pamela Burrage, Thomas A. Mccourt.

3) Reducing the error of Monte Carlo Algorithms by Learning Control Variates. / Brendan Tracey, David H. Wolpert.

4) Latin hypercube sampling with inequality constraints. / Brendan Tracey, David H. Wolpert.

5) Brigo, D. Interest Rate Models - Theory and Practice [Text] : book / D. Brigo, F. Mercurio. – K. : Springer, 2001.

6) Ballotta, L. A Levy process-based framework for the fair valuation of participating life insurance contracts, Insurance: Mathematics and Economics [Text] / L. Ballotta. – M. : 2005. – P. 173–196.

7) Hahn, T. Cuba - a library for multidimensional numerical integration [Text] / T. Hahn. – M. : Computer Physics Communications, 2005. – P. 78–95.

8) An application of Malliavin calculus to Monte Carlo methods in finance [Text] : book / E. Fournie, J.-M. Lasry, J. Lebuchoux, P.-L. Lions, N. Touzi. – M. : Finance & Stochastics, 1999. – P. 391–412.

9) Nahm, T. Error estimation and index refinement for dimension-adaptive sparse grid quadrature with applications to the computation of path integrals [Text] : Master's thesis / T. Nahm. – M. : University Bonn, 2005.

10) Novak, E. Intractability results for integration and discrepancy [Text] : book / E. Novak , H. Wozniakowski ; editing by J. Complexity. – X. : 2001. – P. 388-441.

11) The exponent of discrepancy is at most 1.4778 [Text] : book / Math. Comp. – M. : 1997. – P. 1125-1132.

12) Wang, X. The effective dimension and quasi-Monte Carlo integration [Text] : book / X. Wang , K.-T. Fang ; editing by J. Complexity. – K. 2003. – P. 101-124.

13) Sloan, I. Lattice Methods for Multiple Integration [Text] / I. Sloan, S. Joe // Oxford University Press, New York. – 1994.

14) Papageorgiou, A. Faster evaluation of multidimensional integrals [Text] : book / A. Papageorgiou, J. Traub. – K. : Computers in Phyics, 1997. – P. 574-578.

15) Papageorgiou, A. The Brownian bridge does not offer a consistent advantage in quasi-Monte Carlo integration [Text] : book / A. Papageorgiou, J. Complexity. – K. : 2002. – P. 171-186.

16) Nuyens, D. Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel hilbert spaces [Text] : book / D. Nuyens , R. Cools. – K. : Math Comp., 2006. – P. 903-920.

17) Bulletin, A. Pricing guaranteed life insurance participating policies with annual premiums and surrender option [Text] / A. Bulletin. – M. : 2003. – P. 1–17.

18) Broadie, M. Modelling the surrender conditions in equity-linked life insurance, Insurance: Mathematics and Economics [Text] / M. Broadie. – M. : 2005. – P. 270–296.

19) Bellmann, R. Adaptive Control Processes / R. Bellmann // A Guided Tour. Princeton University Press. – USA : 1961.

20) Brigo, D. Interest Rate Models - Theory and Practice [Text] : book / D. Brigo, F. Mercurio. – K. : Springer, 2001.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
____________д.т.н., проф. Шматков С. І.
«____» ________________20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Братусь Максим Андрійович

1. Тема роботи: Імовірнісні методи побудови пробних вибірок по заданому закону розподілу

керівник роботи _____Угрюмов М.Л. д.т.н. професор____________________________,

затверджені наказом по університету від "___" __________ 20__ року №____________

2. Строк подання студентом роботи____

3. Перелік питань, які потрібно розробити)
1) Проаналізувати існуючі імовірнісні методи побудови пробних вибірок по заданому закону.
2)Вибір моделей та методів для побудови пробних вибірок по заданому закону розподілу.
3)Розробка та верифікація програмного забезпечення для побудови пробних вибірок по заданому закону.
4)Тестування розробленого методу.

4. План роботи

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	Вивчення науково-технічної літератури та джерел в Інтернеті.	28.09.20-09.11.20
2	Аналіз існуючих імовірнісних методів побудови пробних вибірок по заданому закону .	10.11.20-14.12.20
3	Розробка та верифікація імовірнісних методів побудови пробних вибірок по заданому закону .	15.12.20-04.03.21
4	Тестування розробленого методу.	05.03.21-18.03.21
5	Розробка рекомендацій по застосуванню розробленого методу.	19.03.21-10.04.21
6	Презентація дипломної роботи керівнику.	11.04.21-02.05.21
7	Редагування дипломної роботи після перевірки керівника.	03.05.21-17.05.21

5. Дата видачі завдання____________________________________

Студент ___М. А. Братусь___ ________________
ініціали, прізвище підпис

Керівник роботи ___М. Л. Угрюмов___ ________________
ініціали, прізвище підпис

Додаток Б

Затверджую

____

«_____»__________2021 р.

Технічне завдання

на розробку програмного виробу «Математична модель управління якістю вибірки даних»

1.	Введення	Назва: Математична модель управління якістю вибірки даних. Область застосування: Задачі оцінки надійності технічних систем.
2.	Підстава для розробки	1) Навчальний план ФКН за фахом 151 – «Автоматизація та комп'ютерно-інтегровані технології» 2) Завдання на кваліфікаційну роботу затверджено наказом №0210-05/477 від 17.03.2021 представити як Додаток А до пояснювальної записки до кваліфікаційної роботи.
3.	Призначення та цілі розробки	1) Продукт призначений для використання в технічних системах на виробництві. 2) Дослідження, аналіз та порівняння імовірнісних методів побудови пробних вибірок даних.
4.	Вхідні дані для розробки	1) Кількість даних(представлені у вигляді крапок) 2) Математичне очікування та дисперсія нормального закону розподілу 3) Кількість вхідних параметрів (математичне очікування та дисперсія)

5.	Вимоги до програмного продукту (моделі)	Програма повинна: 1) Представляти з себе математичну модель для побудови вибірки даних за усіченим нормальним законом розподілу; 2) Будувати графіки за даними розрахунку алгоритму усіченого нормального розподілу; 3) Програма повинна записувати результати в вихідний файл; Для виконання програми потрібно: 1) Для виконання програми необхідний працюючий ПК; 2) Початкові дані; 3) Операційна система Windows 10; 4) Вимоги до маркування та упаковки (не висуваються); 5) Вимоги до транспортування і зберігання (не висуваються); 6) Спеціальні вимоги (не пред’являються).
6.	Вимоги до програмної документації	Програмною документацією щодо розроблюваного програмного продукту вважати: 1. Справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи); 2. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);

		3. Опис моделі (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи). 4. Лістинг програмного виробу (представити фрагмент як Додаток Г до пояснювальної записки кваліфікаційної роботи).	
7.	Вимоги до техніко-економічних показників	1) Оцінка економічної ефективності – непотрібна. 2) Визначення економічних переваг продукту в порівнянні з вітчизняними та зарубіжними аналогами – виконати в розділі 2.	
8.	Стадії і етапи розробки	Дата	Назва етапу
		від 28 вересня 2020 до 9 листопад 2020	Вивчення науково-технічної літератури та джерел.
		від 10 листопад 2020 до 14 грудня 2020	Аналіз існуючих імовірнісних методів побудови пробних вибірок по заданому закону.
		від 15 грудня 2020 до 4 березня 2021	Розробка та верифікація імовірнісних методів побудови пробних вибірок.
		від 5 березня 2021 до 18 березня 2021	Тестування розробленого методу.
		від 19 березня 2021 до 10 квітня 2021	Розробка рекомендацій по застосуванню продукту.

		від 11 квітня 2021 до 2 травня 2021 від 3 травня 2021 до 17 травня 2021	Презентація дипломної роботи керівнику. Редагування дипломної роботи після перевірки керівника.
9.	Порядок контролю і приймання програмного продукту (моделі)	1) Перевірку ходу розробки програми виконувати раз в 1 тиждень. 2) Випробування програмного продукту провести відповідно до програми та методики випробувань на базі комп'ютерного класу . 3) Захист розробленої моделі провести на засіданні Атестаційної комісії. 4) Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD-R компакт-диску.	

Виконавець
студент групи КУ-41
Братусь М.А
____

Замовник
д.т.н, професор
Угрюмов М.Л.
____

Додаток В

Затверджую

____

«_____»__________2021 р.

ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ
програмного продукту «Математична модель управління якості вибірки даних»

1. Об’єкт випробувань

Об’єктом випробувань є математична модель управління якості вибірки даних.

2. Мета випробування

Перевірка відповідності функціональності програмної реалізації заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Вимоги до програми

Програма повинна:

1) Представляти з себе математичну модель для побудови вибірки даних за усіченим нормальним законом розподілу;
2) Будувати графіки за даними розрахунку алгоритму усіченого нормального розподілу;
3) Програма повинна записувати результати в вихідний файл;

Для виконання програми потрібно:

1) Для виконання програми необхідний працюючий ПК;
2) Початкові дані;
3) Операційна система Windows;
4) Вимоги до маркування та упаковки (не висуваються);
5) Вимоги до транспортування і зберігання (не висуваються);
6) Спеціальні вимоги (не пред’являються).

4. Вимоги до програмної документації

Склад програмної документації, що пред’являється на випробуванні:

Програмною документацією щодо розроблюваного програмного продукту вважати:

1. Справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
2. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
3. Опис моделі (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).
4. Фрагмент лістингу програмного виробу (представити як Додаток Г до пояснювальної записки кваліфікаційної роботи).

5. Засоби випробувань

Необхідна наявність ПК з програмним забезпеченням, необхідним для роботи програмного продукту.

6. Програма і методика випробувань

1. Перевірка програмної документації
   1.1. Перевірка складу програмної документації. Перевірку здійснювати за критерієм наявності, представленої в ТЗ документації.
   1.2. Перевірка якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ГОСТ 19.301-79 ЕСПД. «Програма і методика випробувань».
2. Перевірка працездатності моделі

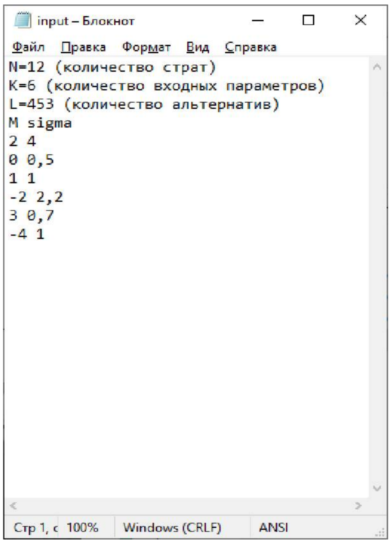


Рисунок В 1 – вписуємо вхідні дані в текстовий документ

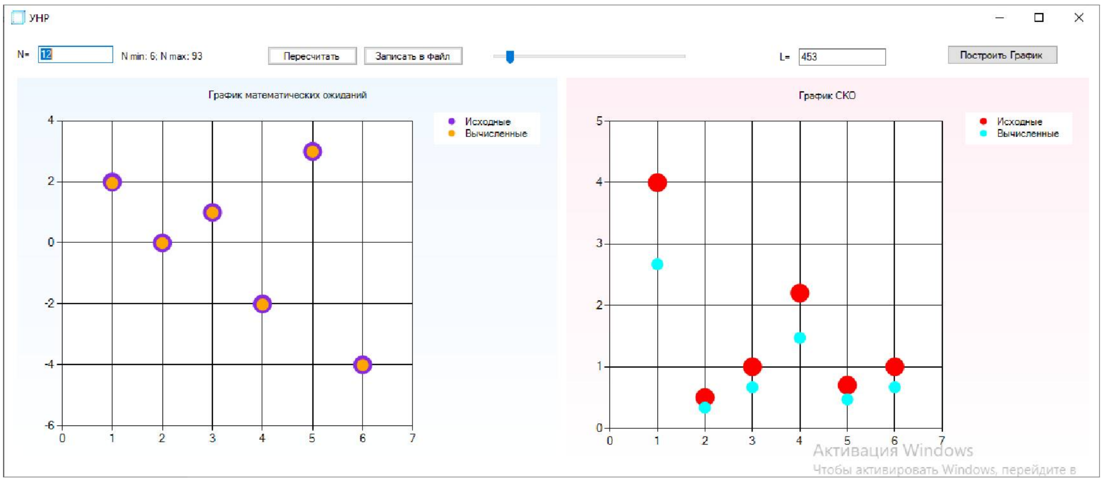
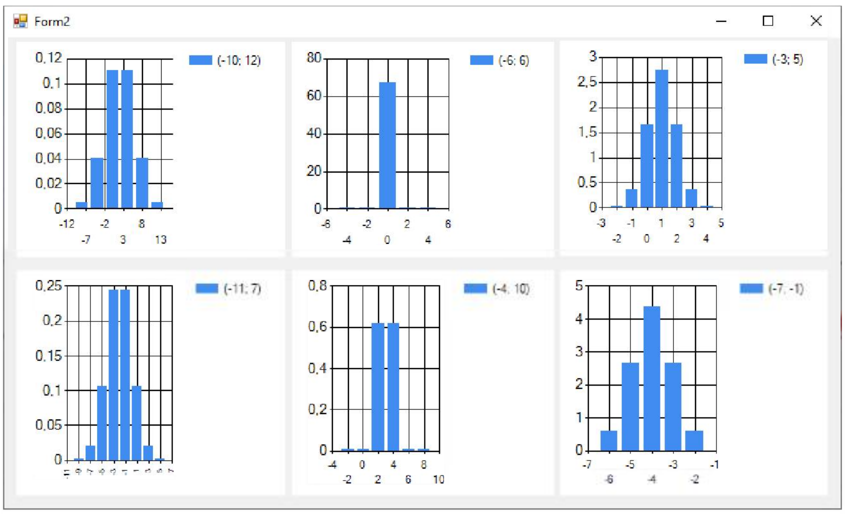


Рисунок В 2 – на моделі ми бачимо що точки відображаються правильно



РисунокВ3 – Побудовані графіки за усіченим нормальним законом розподілу побудувалися правильно з вхідних даних

Висновок до тесту 1: з малюнків в2 та в3 можемо сказати що тест пройшов успішно

Тест 2: Перевірка при інших вхідних даних повинні змінюватися графіки усіченого нормального розподілу.

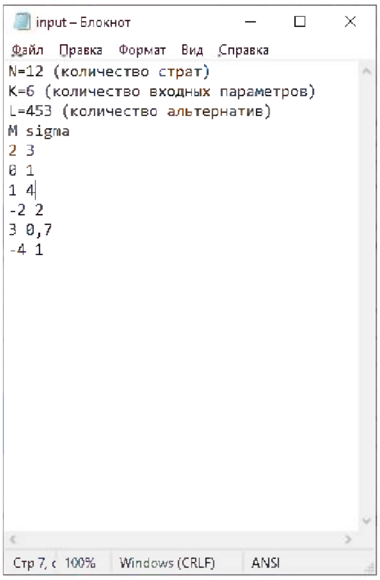


Рисунок В 4 – Змінюємо вхідні данні в текстовому документі

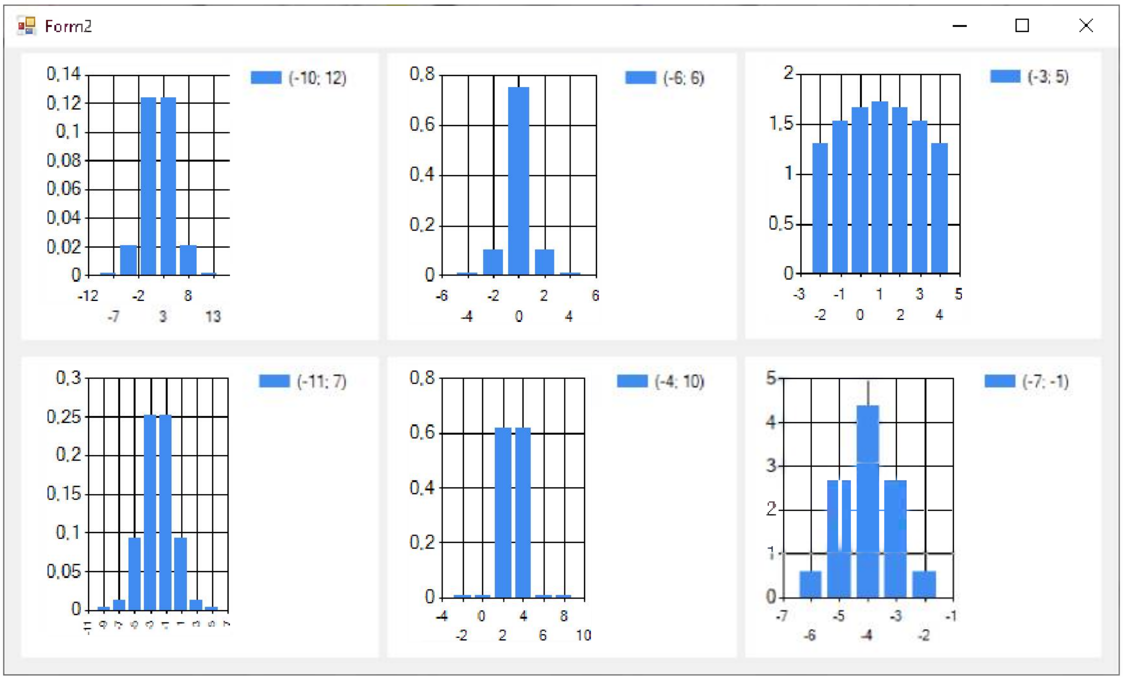


Рисунок В 5 – при зміні вхідних даних змінилися графіки.

Висновок до тесту 2: при інших вхідних даних графіки змінюються, як показано на малюнку В5.

Тест 3: Перевіряємо функціональність робочих елементів в математичній моделі:

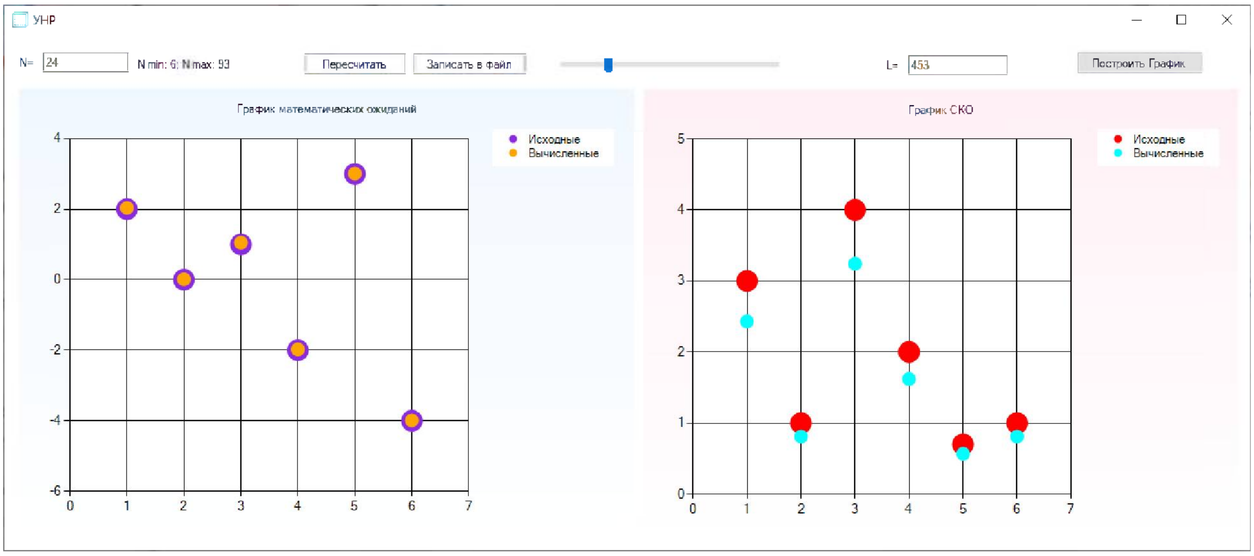


Рисунок В 6 – початкова математична модель без змін

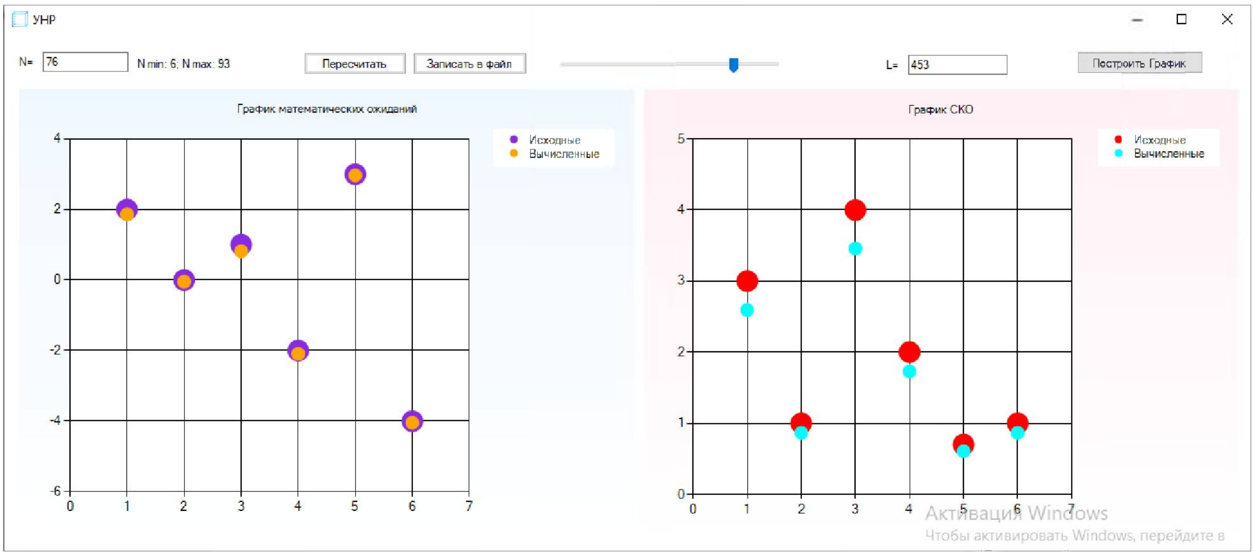


Рисунок В 7 – Змінена математична модель після зміни кількості стратегій та кількості точок.

Висновок до тесту 3: зміна робочих елементів таких, як кількість стратегій та кількість точок в математичній моделі працює правильно.

Висновок: всі три тести, котрі проводилися над програмним виробом пройшли успішно, помилок не було виявлено, отже даний програмний виріб є повністю робочим.

Виконавець: студент групи КУ-41, Братусь Максим.

Додаток Г

Лістинг коду

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace DiplomWinForm
{
    public partial class Form1 : Form
    {
        double[,] laplasTable;
        Random random = new Random();
        int N, Nmin, Nmax;
        int K;
        int L;
        double p;
        M_sigma[] msigma;
        double[] m_out;
        double[] sigma_out;
        Cut[,] cuts;
        double[,] middles;
        double[,] sigma;
        double[,,] vectors;

        struct Cut
        {
            public double a;
            public double b;
        }

        public Form1()
        {
            InitializeComponent();
            readFile();
            doAlgorithm();
            drawChart();
            writeFile();
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            recalculate();
        }

        private void recalculate()
        {
            N = int.Parse(textBox1.Text);
            if (N < Nmin)
            {
                N = Nmin;
            }
            if (N > Nmax)
            {
                N = Nmax;
            }
            L = int.Parse(textBox2.Text);
            //if (L > N) L = N;
            textBox2.Text = L.ToString();
            textBox1.Text = N.ToString();
            trackBar1.Value = N;
            p = 1.0 / N;
            doAlgorithm();
            drawChart();
        }

        private void drawChart()
        {
            this.chart1.Series.Clear();
            this.chart1.Titles.Clear();

            this.chart1.Titles.Add("График математических ожиданий");

            Series series = this.chart1.Series.Add("Исходные");
            this.chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            this.chart1.Series[0].Color = Color.BlueViolet;
            this.chart1.Series[0].MarkerStyle = MarkerStyle.Circle;
            this.chart1.Series[0].MarkerSize = 22;
            for (int i=0; i < msigma.Length; i++)
            {
                series.Points.AddXY(i + 1, msigma[i].M);
            }

            Series series2 = this.chart1.Series.Add("Вычисленные");
            this.chart1.Series[1].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            this.chart1.Series[1].Color = Color.Orange;
            this.chart1.Series[1].MarkerStyle = MarkerStyle.Circle;
            this.chart1.Series[1].MarkerSize = 14;
```

```
            for (int i = 0; i < m_out.Length; i++)
            {
                series2.Points.AddXY(i + 1, m_out[i]);
            }

            this.chart2.Series.Clear();
            this.chart2.Titles.Clear();
            this.chart2.Titles.Add("График СКО");

            Series series3 = this.chart2.Series.Add("Исходные");
            this.chart2.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            this.chart2.Series[0].Color = Color.Red;
            this.chart2.Series[0].MarkerStyle = MarkerStyle.Circle;
            this.chart2.Series[0].MarkerSize = 22;
            for (int i = 0; i < msigma.Length; i++)
            {
                series3.Points.AddXY(i + 1, msigma[i].sigma);
            }

            Series series4 = this.chart2.Series.Add("Вычисленные");
            this.chart2.Series[1].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            this.chart2.Series[1].Color = Color.Aqua;
            this.chart2.Series[1].MarkerStyle = MarkerStyle.Circle;
            this.chart2.Series[1].MarkerSize = 14;
            for (int i = 0; i < sigma_out.Length; i++)
            {
                series4.Points.AddXY(i + 1, sigma_out[i]);
            }
        }

        private void readFile()
        {
            //считывание таблицы лапласа
            string[] laplasFile = File.ReadAllLines("laplas.txt");
            laplasTable = new double[laplasFile.Length, 2];
            for (int i = 0; i < laplasFile.Length; i++)
            {
                string[] lineParts = laplasFile[i].Split(' ');
                laplasTable[i, 0] = double.Parse(lineParts[0]);
                laplasTable[i, 1] = double.Parse(lineParts[1]);
            }

            //считывание входных данных
            string[] inputFile = File.ReadAllLines("input.txt");
            string nstring = inputFile[0].Remove(0, 2);
            string[] nstringmass = nstring.Split(' ');
            N = int.Parse(nstringmass[0]);
            nstring = inputFile[1].Remove(0, 2);
```

```
    nstringmass = nstring.Split(' ');
    K = int.Parse(nstringmass[0]);
    nstring = inputFile[2].Remove(0, 2);
    nstringmass = nstring.Split(' ');
    L = int.Parse(nstringmass[0]);
    Nmin = K;
    if (N < Nmin) N = Nmin;
    msigma = new M_sigma[K];
    for (int i = 4; i < 4+K; i++)
    {
        string[] lineParts = inputFile[i].Split(' ');
        msigma[i - 4] = new M_sigma(double.Parse(lineParts[0]), double.Parse(lineParts[1]));
    }
    Nmax = 10000;
    Nmax = getMaxN(N);
    if (N > Nmax) N = Nmax;
    p = 1.0 / N;
    textBox1.Text = N.ToString();
    textBox2.Text = L.ToString();
    trackBar1.Minimum = Nmin;
    trackBar1.Maximum = Nmax;
    trackBar1.Value = N;
    label3.Text = String.Format("N min: {0}; N max: {1}", Nmin, Nmax);
}

private void writeFile()
{
    StreamWriter sw = new StreamWriter("out.txt");

    //шаг 1
    for (int i = 0; i < cuts.Length / N; i++)
    {
        sw.Write("Отрезки для пары №{0}: ", i + 1);
        for (int j = 0; j < N; j++)
        {
            string a = Math.Round(cuts[i, j].a, 5).ToString();
            string b = Math.Round(cuts[i, j].b, 5).ToString();
            sw.Write("({0}, {1}) ", a, b);
        }
        sw.WriteLine();
    }
    sw.WriteLine("\n\n\n");

    //шаг 2
    /*
    for (int i = 0; i < K; i++)
    {
        sw.WriteLine("Пара номер: {0}", i + 1);
        for (int j = 0; j < K; j++)
        {
```

```
        sw.Write("(");
        for (int l = 0; l < N; l++)
        {
            sw.WriteLine(" {0}", vectors[i, j, l]);
        }
        sw.WriteLine(")");
    }
    sw.WriteLine();
}
*/

//шаг 3
sw.Write("Альт.(№) ");
for (int i = 0; i < K; i++)
{
    sw.Write(" | X{0,-10:0}", i + 1);
}
sw.WriteLine();
for (int i = 0; i < L; i++)
{
    sw.WriteLine(new String('_', 10 + (14 * K)));
    sw.Write("M{0,-8:0}", i + 1);
    for (int j = 0; j < K; j++)
    {
        sw.Write(" | {0,-10:0.00000} ", middles[j, i]);
    }
    sw.WriteLine();
}
sw.WriteLine(new String('_', 10 + (14 * K)));
sw.Write("{0,-9:0}", "M(сред.)");
for (int i=0; i<m_out.Length; i++)
{
    sw.Write(" | {0,-10:0.00000} ", m_out[i]);
}
sw.WriteLine("\n\n\n");

sw.Write("Альт.(№) ");
for (int i = 0; i < K; i++)
{
    sw.Write(" | X{0,-10:0}", i + 1);
}
sw.WriteLine();
for (int i = 0; i < L; i++)
{
    sw.WriteLine(new String('_', 10 + (14 * K)));
    sw.Write("s{0,-8:0}", i + 1);
    for (int j = 0; j < K; j++)
    {
        sw.Write(" | {0,-10:0.00000} ", sigma[j, i]);
    }
```

```
            sw.WriteLine();
        }
        sw.WriteLine(new String('_', 10 + (14 * K)));
        sw.Write("{0,-9:0}", "s(сред.)");
        for (int i = 0; i < sigma_out.Length; i++)
        {
            sw.Write(" | {0,-10:0.00000} ", sigma_out[i]);
        }

        sw.Close();
    }

    private void doAlgorithm()
    {
        //шаг 1
        cuts = getCuts();

        //шаг 2
        vectors = getVectors();

        //шаг 3
        middles = new double[K, L];
        sigma = new double[K, L];
        for (int i = 0; i < K; i++)
        {
            for (int j = 0; j < L; j++)
            {
                double middle = 0;
                for (int l = 1; l < N - 1; l++)
                {
                    middle += vectors[i, j, l];
                }
                middle = middle / (N - 2);
                middles[i, j] = middle;
            }
        }
        for (int i = 0; i < K; i++)
        {
            for (int j = 0; j < L; j++)
            {
                double x = 0;
                for (int l = 1; l < N - 1; l++)
                {
                    x += Math.Pow((vectors[i, j, l] - middles[i, j]),2);
                }
                x = Math.Sqrt(x / (N - 1));
                sigma[i, j] = x;
            }
        }
```

```
    //шаг 4
    m_out = new double[K];
    sigma_out = new double[K];
    for (int i = 0; i < K; i++)
    {
        double m = 0;
        double s = 0;
        for (int j = 0; j < L; j++)
        {
            m += middles[i, j];
            s += sigma[i, j];
        }
        m_out[i] = m / L;
        sigma_out[i] = s / L;
    }

}

private Cut[,] getCuts()
{
    Cut[,] output = new Cut[msigma.Length, N];
    for (int i = 0; i < msigma.Length; i++)
    {
        double a = double.MinValue;

        for (int j = 0; j < N; j++)
        {
            double x1 = p + Fx((a - msigma[i].M) / msigma[i].sigma);
            double x2 = Xf(x1);
            double b = msigma[i].sigma * x2 + msigma[i].M;
            output[i, j].a = a;
            output[i, j].b = b;
            a = b;
        }

        output[i, N - 1].b = double.MaxValue;
    }
    return output;
}

private int getMaxN(int n)
{
    double p = 1.0 / n;
    for (int i = 0; i < msigma.Length; i++)
    {
        double a = double.MinValue;

        for (int j = 0; j < n; j++)
        {
            double x1 = p + Fx((a - msigma[i].M) / msigma[i].sigma);
```

```
            double x2 = Xf(x1);
            double b = msigma[i].sigma * x2 + msigma[i].M;
            if ((a == b) && (j!=n-1))
            {
                return n - 1;
            }
            a = b;
        }
    }
    if (n == 10000) return n;
    return getMaxN(n+1);
}

private double Fx(double x)
{
    double fx = -1;
    double lastMin = 0;
    double xMod = Math.Abs(x);

    for (int i = 0; i < laplasTable.Length / 2; i++)
    {
        if (laplasTable[i, 0] < xMod)
        {
            lastMin = laplasTable[i, 1];
            if (i + 1 < laplasTable.Length / 2)
            {
                double lastMinDelta = Math.Abs(laplasTable[i, 0] - xMod);
                double lastMaxDelta = Math.Abs(laplasTable[i + 1, 0] - xMod);
                if (lastMinDelta <= lastMaxDelta)
                {
                    lastMin = laplasTable[i, 1];
                }
                else
                {
                    lastMin = laplasTable[i + 1, 1];
                }
            }
        }
        if (laplasTable[i, 0] == xMod)
        {
            fx = laplasTable[i, 1];
        }
    }
    if (fx == -1) fx = lastMin;
    if (x < 0) fx = -1 * fx;
    return fx;
}

private double Xf(double f)
{
```

```
    double xf = -1;
    double lastMin = 0;
    double fMod = Math.Abs(f);
    for (int i = 0; i < laplasTable.Length / 2; i++)
    {
        if (laplasTable[i, 1] < fMod)
        {
            lastMin = laplasTable[i, 0];
            if (i + 1 < laplasTable.Length / 2)
            {
                double lastMinDelta = Math.Abs(laplasTable[i, 1] - fMod);
                double lastMaxDelta = Math.Abs(laplasTable[i + 1, 1] - fMod);
                if (lastMinDelta <= lastMaxDelta)
                {
                    lastMin = laplasTable[i, 0];
                }
                else
                {
                    lastMin = laplasTable[i + 1, 0];
                }
            }
        }
        if (laplasTable[i, 1] == fMod)
        {
            xf = laplasTable[i, 0];
        }
    }
    if (xf == -1) xf = lastMin;
    if (f < 0) xf = -1 * xf;
    return xf;
}

private double[,,] getVectors()
{
    double[,,] result = new double[K, L, N];

    for (int i = 0; i < K; i++)
    {
        List<double> used = new List<double>();
        bool found = false;
        for (int j = 0; j < N; j++)
        {
            for (int l = 0; l < L; l++)
            {
                found = false;
                do
                {
                    //cuts = cuts;
                    double num = getRandom(cuts[i, j].a, cuts[i, j].b);
                    if (cuts[i,j].a == cuts[i,j].b)
```

```
                {
                    result[i, l, j] = num;
                    found = true;
                }
                else if (!isUsed(num, used, cuts[i, j]))
                {
                    used.Add(num);
                    result[i, l, j] = num;
                    found = true;
                }
            }
            while (!found);
        }
        found = false;
    }
}

return result;
}

private double getRandom(double min, double max)
{
    return min + random.NextDouble() * (max - min);
}

private bool isUsed(double num, List<double> usedNums, Cut cut)
{
    bool isUsed = false;
    if ((usedNums.IndexOf(num) >= 0) || (num == cut.a) || (num == cut.b)) isUsed = true;
    return isUsed;
}

private void trackBar1_ValueChanged(object sender, EventArgs e)
{
    textBox1.Text = trackBar1.Value.ToString();
    recalculate();
}

private void textBox2_TextChanged(object sender, EventArgs e)
{

}

private void button3_Click(object sender, EventArgs e)
{
    Form2 form = new Form2();
    form.Show();
}

private void Form1_Load(object sender, EventArgs e)
```

```
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            writeFile();
        }
    }

    class M_sigma
    {
        public double M;
        public double sigma;

        public M_sigma(double M, double sigma)
        {
            this.M = M;
            this.sigma = sigma;
        }
    }
}
using System;
using System.IO;
using System.Windows.Forms;

namespace DiplomWinForm
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            using (StreamReader sr = new StreamReader("input.txt",
System.Text.Encoding.Default))
            {
                string line;
                int c = 0;
                while ((line = sr.ReadLine()) != null)
                {
                    c++;
                    if (c >= 5)
                    {
                        string[] data = line.Split(' ');
                        if (c == 5)
                            drawChart1(Double.Parse(data[0]), Double.Parse(data[1]));
```

```
                else if (c == 6)
                    drawChart2(Double.Parse(data[0]), Double.Parse(data[1]));
                else if (c == 7)
                    drawChart3(Double.Parse(data[0]), Double.Parse(data[1]));
                else if (c == 8)
                    drawChart4(Double.Parse(data[0]), Double.Parse(data[1]));
                else if (c == 9)
                    drawChart5(Double.Parse(data[0]), Double.Parse(data[1]));
                else if (c == 10)
                    drawChart6(Double.Parse(data[0]), Double.Parse(data[1]));
            }
        }
    }
}

private void drawChart1(double m, double q)
{
    chart1.Series[0].LegendText = "(-10; 12)";
    for (double i = -8; i <= 12; i+= 4)
    {
        chart1.Series[0].Points.AddXY(i, c(-6, 6, m, q) * f(i, m, q));
    }
}
private void drawChart2(double m, double q)
{
    chart2.Series[0].LegendText = "(-6; 6)";
    for (double i = -4; i <= 4; i += 2)
    {
        chart2.Series[0].Points.AddXY(i, c(-3, 3, m, q) * f(i, m, q));
    }
}
private void drawChart3(double m, double q)
{
    chart3.Series[0].LegendText = "(-3; 5)";
    for (double i = -2; i <= 4; i += 1)
    {
        chart3.Series[0].Points.AddXY(i, c(-0.3, 0.3, m, q) * f(i, m, q));
    }
}
private void drawChart4(double m, double q)
{
    chart4.Series[0].LegendText = "(-11; 7)";
    for (double i = -9; i <= 5; i += 2)
    {
        chart4.Series[0].Points.AddXY(i, c(-3, 3, m, q) * f(i, m, q));
    }
}
private void drawChart5(double m, double q)
{
    chart5.Series[0].LegendText = "(-4; 10)";
```

```
            for (double i = -2; i <= 8; i += 2)
            {
                chart5.Series[0].Points.AddXY(i, c(-3, 3, m, q) * f(i, m, q));
            }
        }
        private void drawChart6(double m, double q)
        {
            chart6.Series[0].LegendText = "(-7; -1)";

            for (double i = -6; i <= -2; i++)
            {
                chart6.Series[0].Points.AddXY(i, c(-3, 3, m, q) * f(i, m, q));
            }
        }

        private double f(double x, double m, double q)
        {
            return (1 / (q * Math.Sqrt(2*Math.PI))) * Math.Exp(-Math.Pow(x - m, 2) / (2 * q * q));
        }

        private double c(double x1, double x2, double m, double q) => 1 / df(x1, x2, m, q);

        private double df(double x1, double x2, double m, double q)
        {
            return ((x2 - x1) / 2) * (f((x1 + x2) / 2 - (x2 - x1) / (2 * Math.Sqrt(3)), m, q) + (f((x1 + x2)
/ 2 + (x2 - x1) / (2 * Math.Sqrt(3)), m, q)));
        }
    }
}
```