МІНІСТЕРСТВО ОСВІТИІ НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедрою _____
д.е.н., доцент Левицький С.І.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

РОЗРОБКА СИСТЕМИ ГЕНЕРАЦІЇ ВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ АНАЛІЗУ АКУСТИЧНИХ СИГНАЛІВ

Виконала
ст. гр. ІПЗ-212/м _____ Г.В. Куликівська
(підпис)

Керівник
к.т.н. _____ Д.Г. Медведев
(підпис)

Запоріжжя
2023

ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. кафедрою

_____
д.е.н., доцент Левицький С.І.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ

Студенту гр.ІПЗ-212/м, спеціальності *«Інженерія програмного забезпечення»*

Куликівська Ганна Вікторівна

1.Тема: Розробка системи генерації випадкових чисел на основі аналізу акустичних сигналів

затверджена наказом по інституту «___» _____ 20__ р № ___

2. Термін здачі студентом закінченої роботи: «___» _____ 2023 р

3. Перелік питань, що підлягають розробці:

1. Аналіз існуючих методів та підходів до генерації випадкових чисел
2. Властивості випадкових чисел
3. Методи аналізу акустичних сигналів
4. Моделювання акустичних сигналів

5. Застосування акустичного аналізу
6. Розробка алгоритму роботи системи.
7. Розробка архітектури системи
8. Тестування системи
9. Огляд генерованих послідовностей випадкових чисел

4. Календарний графік підготовки кваліфікаційної роботи

№ етапу	Зміст	**Терміни виконання**	**Готовність етапу, %**	**Підпис керівника про повну готовність етапу, дата**
1.	Корегування теми магістерської кваліфікаційної роботи, збір практичного матеріалу за темою кваліфікаційної магістерської роботи	**04.09.23-17.10.23**		
2.	**І атестація** І розділ кваліфікаційної магістерської роботи	**23.10.23-28.10.23**		
3.	**ІІ атестація** ІІ розділ кваліфікаційної магістерської роботи	**20.11.23-25.11.23**		
4.	**ІІІ атестація** ІІІ розділ кваліфікаційної магістерської роботи, висновки та рекомендації, додатки, реферат, перевірка програмою «Антиплагіат»	**18.12.23-23.12.23**		
5.	Доопрацювання кваліфікаційної магістерської роботи, підготовка презентації, отримання відгуку керівника і рецензії	**25.12.23-06.01.24**		
6.	**Попередній захист кваліфікаційної магістерської роботи**	**08.01.24-13.01.24**		
7.	Подача кваліфікаційної магістерської роботи на кафедру	за 3 дні до захисту		
8.	**Захист кваліфікаційної магістерської роботи**	**15.01.24-20.01.24**		

Дата видачі завдання

Керівник _____ Д.Г. Медведев «___» _____ 2024

Студент _____ Г.В.Куликівська «___» _____ 2024

РЕФЕРАТ

Кваліфікаційна магістерська робота містить 91 стор., 14 рис., 1 додаток, 23 використаних джерела.

Об’єкт роботи: системи генерації випадкових чисел.

Предмет роботи: система для генерації випадкових чисел на основі аналізу акустичних сигналів та дослідження її аспектів.

Мета роботи: розробка системи генерації випадкових чисел, яка базується на аналізі акустичних сигналів та демонстрація її ефективності.

Для досягнення даної мети були поставлені наступні завдання: аналіз існуючих методів та підходів до генерації випадкових чисел; вивчення властивостей випадкових чисел; дослідження різних методів аналізу акустичних сигналів; розгляд теоретичних моделей акустичних сигналів; створення алгоритму, який базується на аналізі акустичних сигналів та генерує випадкові числа; розробка архітектури системи; виконання тестів для перевірки роботи системи та налагодження її роботи.

У дослідженні була розроблена та протестована система генерації випадкових чисел на основі аналізу акустичних сигналів. Результати роботи свідчать про успішність обраного підходу до генерації випадкових чисел і підтверджують його потенціал для застосування. Таким чином, система відкриває нові можливості для забезпечення високого рівня випадковості у різних застосуваннях, і може стати важливим інструментом у сучасному технологічному розвитку.

ВИПАДКОВІ ЧИСЛА, АКУСТИЧНИЙ СИГНАЛ, РІВНОМІРНИЙ РОЗПОДІЛ, СТАТИСТИЧНА СТІЙКІСТЬ, ІНФОРМАЦІЙНА БЕЗПЕКА, КРИПТОГРАФІЯ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення	Умови використання
F		
Fast Fourier Transform	FFT	
L		
Лінійні конгруентні генератори	LCG	
M		
Mel-Frequency Cepstral Coefficients	MFCC	
P		
Pseudo-Random Number	PRN	
Pseudorandom Number Generator	PRNG	
Q		
Quantum Random Number Generator	QRNG	
R		
Random Number Generator	RNG	
T		
True Random Number Generator	TRNG	

ВСТУП

Генерація випадкових чисел є ключовою складовою в багатьох аспектах сучасного життя, від криптографії до наукового моделювання і статистики. Забезпечення високої якості та непередбачуваності випадкових чисел є важливою задачею, оскільки безпека, точність і надійність багатьох систем і додатків залежать від якості генерації випадковості. З огляду на це вчені та інженери намагаються постійно розвивати нові підходи та методи для генерації випадкових чисел, які відповідали б сучасним вимогам і викликам.

Об'єктом дослідження магістерської роботи є системи генерації випадкових чисел, а предметом дослідження є система, що ґрунтується на аналізі акустичних сигналів для створення випадкових чисел. Цей новаторський підхід використовує можливості акустичних сигналів як джерела випадковості та пропонує інноваційний метод генерації випадковості, який може стати ключовим фактором у різних областях.

Метою дослідження є розробка та вивчення системи генерації випадкових чисел, яка базується на аналізі акустичних сигналів. Для досягнення цієї мети визначено низку завдань:

1. Аналіз існуючих методів та підходів до генерації випадкових чисел.
2. Дослідження властивостей випадкових чисел, включаючи рівномірність розподілу, незалежність та статистичну стійкість.
3. Вивчення методів аналізу акустичних сигналів.
4. Розгляд теоретичних моделей акустичних сигналів та їх використання для розуміння процесу генерації акустичних даних.
5. Розробка алгоритму для генерації випадкових чисел на основі аналізу акустичних сигналів.
6. Розробка архітектури системи, включаючи блоки обробки сигналів, аналізу та генерації випадкових чисел.

7. Виконання тестів для перевірки роботи системи та налагодження її функціональності.
8. Візуалізація генерованих послідовностей чисел для оцінки їхньої якості та властивостей.

Результати цього дослідження можуть суттєво сприяти розвитку систем генерації випадковості та підвищенню надійності та безпеки інформації в сучасному світі.

РОЗДІЛ 1
ТЕОРІЯ ТА ПРАКТИКА ГЕНЕРАЦІЇ ВИПАДКОВИХ ЧИСЕЛ

1.1 Вступ в генерацію випадкових чисел

Випадкові числа відіграють ключову роль у цифровому світі, надаючи непередбачуваність та надійність різним аспектам нашого життя та наукових досліджень. Випадкові числа – це числа або послідовності чисел, які виникають без визначеного порядку чи закономірності. Вони є результатом непередбачуваних або стохастичних процесів, і їх характеристики не піддаються передбаченню. Випадкові числа можуть виникати внаслідок різних фізичних явищ, таких як квантові події, шумові коливання, теплові флуктуації тощо.

Ці числа можуть бути використані для моделювання непередбачуваних сценаріїв, створення шифрів, статистичного аналізу, симуляцій, генерації випадкових подій, рандомізації, та інших завдань, де важлива випадковість та відсутність будь-якого впорядкованого шаблону. Випадкові числа мають різноманітні застосування в науці, технологіях та мистецтві, і вони важливі для забезпечення надійності та об'єктивності у великому спектрі областей.

Для створення випадкових чисел у цифровому середовищі використовуються генератори випадкових чисел (RNG). Існують два основних типи RNG:

1) Псевдовипадкові генератори випадкових чисел (PRNG) [9] – це програмні або апаратні засоби, які генерують послідовності чисел, що, на перший погляд, схожі на випадкові числа, але насправді є результатом визначеного математичного алгоритму. Вони називаються «псевдовипадковими», оскільки вони зазвичай базуються на початковому значенні і генерують послідовності чисел, які можуть бути передбаченими, якщо відомо початкове значення.

Основні характеристики PRNG включають:

– початкове значення, яке ініціює генерацію випадкової послідовності. Те саме початкове значення завжди призводить до тієї самої послідовності чисел.

– періодичність: PRNG мають обмежений період, після якого послідовність чисел починає повторюватися. Довжина цього періоду залежить від конкретного алгоритму та використовуваного початкового значення.

– детермінованість: послідовність чисел, що генерується PRNG, є детермінованою та повторюється, коли використовується те саме початкове значення. Це означає, що вони не є істинно випадковими.

– відсутність криптографічної стійкості: PRNG зазвичай не відповідають вимогам криптографічної стійкості та не підходять для застосувань, де важливі висока ступінь випадковості та надійний захист від атак. Оскільки їхні властивості є детермінованими, вони можуть бути передбачені та використані зловмисниками для злому криптографічних захистів.

Попри їхню детермінованість та обмежену випадковість, псевдовипадкові генератори є корисними для багатьох застосувань, де важлива простота та відтворюваність послідовностей чисел, таких як моделювання, статистичний аналіз та випробування програмного забезпечення. Вони широко використовуються в програмуванні та інформатиці для генерації випадкових даних та симуляцій.

2) Генератори справжніх випадкових чисел (TRNG) – це пристрої або процеси, які створюють послідовності чисел, що вважаються істинно випадковими. Вони відрізняються від PRNG оскільки базуються на непередбачуваних фізичних явищах, таких як квантові процеси, радіоактивний розпад, електромагнітний шум чи інші подібні інциденти. TRNG володіють властивістю справжньої випадковості, що робить їх

придатними для високочутливих застосувань, де надійність та непередбачуваність є обов’язковими параметрами для генерування.

Основні характеристики TRNG включають:

– фізичне джерело випадковості: TRNG використовують фізичні явища як джерело випадковості. Це може бути квантовий процес, який непередбачувий за своєю природою, або інше надійне фізичне явище.

– відсутність детермінованості: використання справжніх випадкових процесів гарантує відсутність будь-якої детермінованості або повторюваності в генерованих числах. Їхні значення неможливо передбачити заздалегідь.

– висока ступінь випадковості: результати, отримані з використанням TRNG, вважаються вкрай випадковими та надійними, що робить їх ідеальними для застосувань, де важлива висока ступінь випадковості, такі як криптографія та критичні системи.

– спеціалізоване обладнання: TRNG можуть вимагати спеціалізованого обладнання для спостереження та реєстрації фізичних процесів, що використовуються для генерації випадковості.

TRNG зазвичай використовуються в критичних додатках, таких як криптографічні системи, де надійність та випадковість великої важливості для захисту інформації. Також вони знаходять своє використання у наукових дослідженнях, статистиці, генетиці, космічних дослідженнях та інших областях, де важлива справжня випадковість.

Випадкові числа, створені засобами RNG грають критичну роль у різних галузях науки, технології та практичного життя [15]. Наприклад:

– Криптографія: випадкові числа є фундаментальним елементом у створенні безпечних шифрів та ключів. Вони гарантують, що криптографічні системи залишаються надійними та стійкими до атак.

– Інформаційна безпека: у сучасному світі збереження конфіденційної інформації вимагає використання випадкових чисел для створення паролів, ключів та інших ідентифікаторів безпеки.

– Симуляції: випадкові числа допомагають в симуляціях та моделях, від вивчення кліматичних змін до економічних прогнозів та поведінки систем.

– Медицина та генетика: в дослідженнях генетики та медицини випадкові числа використовуються для визначення статистичних зв'язків, розподілу лікарських тестів та генетичних аналізів.

– Фінанси: в управлінні ризиками, аналізі фінансових ринків та прогнозуванні цін випадкові числа використовуються для моделювання невизначеності та варіабельності.

– Геоінформаційні системи: випадкові числа використовуються для генерації картографічних даних, обробки сигналів та маршрутизації.

– Інтернет та мережева безпека: випадкові числа грають важливу роль у процесі ідентифікації та забезпеченні безпеки мережевого зв'язку, захищаючи дані від несанкціонованого доступу.

– Ігри та розваги: RNG створюють непередбачуваність та різноманітність, що робить гру більш захопливою

Випадкові числа є важливим ресурсом, який допомагає моделювати невизначеність та випадкові події в різних аспектах нашого життя та допомагають у забезпеченні безпеки, точності та непередбачуваності в різних галузях.

1.2 Класичні методи генерації PRN

Класичні методи генерації псевдовипадкових чисел (PRN) використовуються для отримання числових послідовностей, які, хоча виглядають випадковими, фактично генеруються за допомогою детермінованих алгоритмів. Вони широко використовуються в різних

областях, таких як комп'ютерна графіка, моделювання, статистика та криптографія. Розглянемо декілька класичних методів генерації PRN:

1) Лінійні конгруентні генератори (LCG)[3] – це одна з найпоширеніших та найпростіших форм PRNG. Вони використовуються для генерації послідовностей чисел, які, на перший погляд, схожі на випадкові, але фактично генеруються за допомогою детермінованого математичного алгоритму.

Основними складовими LCG є:

– Початкове значення (сід): LCG потребує початкового значення, відомого як «сід». Сід визначає початкову точку генерації. Якщо використовується однаковий сід, то завжди отримується однакова послідовність чисел.

– Множник (a) – це константа, яка впливає на зміну числа в кожному кроці генерації. Вибір правильного множника важливий, оскільки від нього залежить якість та періодичність генерованої послідовності.

– Приріст (c) – це інша константа, яка додається до числа в кожному кроці. Використовується для створення зміщення в генерованій послідовності.

– Модуль (m) – це константа, яка обмежує значення чисел в послідовності. Всі числа в послідовності LCG повинні бути менше за m.

Принцип роботи LCG полягає в обчисленні нового числа X_{n+1} за попереднім значенням X_n за допомогою наступного виразу:

$$X_{n+1} = (aX_n + c)\ mod\ m, \qquad (1)$$

де:

X_{n+1}– нове згенероване число,

X_n– попереднє згенероване число або початкове значення при n = 0,

a – множник,

c – приріст,

m – модуль.

LCG мають кілька переваг, такі як простота та швидкість генерації чисел. Проте вони також мають обмеження. Якщо параметри не вибрані правильно, послідовність може стати передбачуваною і періодичною, що не підходить для деяких криптографічних або симуляційних застосувань. Тому важливо обирати параметри LCG ретельно, а також розглядати інші, більш складні генератори для застосувань, де важлива висока ступінь випадковості.

2) Генератори на основі мережі Фейстеля (або просто «метод Фейстеля»)[8, 21] – це підхід до генерації PRN, який використовує структуру мережі Фейстеля, що виникла в криптографії, але була застосована й для генерації випадкових чисел.

Основні кроки методу Фейстеля:

– Початкове значення (сід): Генератор отримує початкове значення, відоме як «сід» (seed), як вхідні дані для початку ітерацій.

– Розбиття на блоки: сід поділяється на два блоки (лівий і правий), які стають початковими значеннями для першої ітерації.

– Ітерація: проводиться ітеративний процес, в якому лівий та правий блоки піддаються операціям змішування і перетворення. Це може включати математичні операції, такі як додавання, віднімання, побітові операції і т.д. Результат цих операцій утворює нові значення для лівого і правого блоків.

– Об'єднання і вивід: нові значення лівого та правого блоків об'єднуються для утворення одного вихідного числа, яке вважається псевдовипадковим.

– Повторення: процес ітерації повторюється для генерації інших псевдовипадкових чисел. Якщо потрібно, ключ або параметри можуть бути змінені для отримання різних послідовностей чисел.

– Контроль параметрів: важливо правильно налаштувати параметри, такі як функції перетворення і ключі, для забезпечення властивостей псевдовипадковості та безпеки.

Метод Фейстеля дозволяє створювати великі обсяги PRN, використовуючи складний ітеративний процес. Він може бути корисним в різних застосуваннях, включаючи криптографію, симуляції та статистичний аналіз. Однак важливо пам'ятати про належну конфігурацію та безпеку параметрів для запобігання передбачуваності в генерованих числах.

3) Метод середнього квадрату[10] полягає в тому, щоб брати квадрат числа, потім вибирати середні розряди, і використовувати їх як нове випадкове число. Принцип полягає у наступних кроках: обирається початкове число, це число квадратується, а отриманий результат поділяється на розряди; для створення нового числа обираються певні розряди з результату квадратування; отримане число стає наступним початковим значенням для наступної ітерації.

Цей метод, хоча й простий, може мати обмежену випадковість та потенційно призводити до виникнення коротких періодів та повторів послідовностей чисел. Неправильний вибір початкового числа може призвести до менш випадкових результатів. У сучасних застосуваннях криптографії цей метод застосовується рідко через його обмеженість та недоліки.

4) Метод обертання із масштабуванням[1]– це техніка, яка використовується для генерування випадкових чисел або для створення псевдовипадкових послідовностей на основі початкового значення, яке змінюється шляхом обертання та масштабування. Цей метод базується на використанні математичних операцій обертання (ротації) та масштабування. Щоб генерувати нові випадкові числа, вхідне значення змінюється шляхом його обертання та масштабування.

Основна ідея полягає в тому, щоб взяти вихідне значення (наприклад, попереднє випадкове число або початкове значення), змінити його шляхом математичних операцій обертання на певний кут та масштабування цього значення на певний коефіцієнт. Ці операції дозволяють створити нове випадкове або псевдовипадкове число.

Наприклад, якщо є випадкове число x, його можна обернути на певний кут (використовуючи тригонометричні функції, такі як синус та косинус) і потім масштабувати його, додавши або множачи його на певний коефіцієнт. Цей новий результат може використовуватися як наступне випадкове число у послідовності.

5) Метод Блюма-Мікелсона [2]– це метод генерації псевдовипадкових чисел, розроблений Джейсоном Блюмом та Майклом Мікелсоном. Цей метод базується на комбінації принципу багатоваріантних генераторів та здатності криптографічних хеш-функцій створювати відомі, але обчислювально важкі проблеми. До основних принципів методу Блюма-Мікелсона входять багатоваріантні генератори (використання декількох незалежних PRNG) та криптографічні хеш-функції (використання криптографічних хеш-функцій, які є важкими для зворотного обчислення, забезпечуючи безпеку та випадковість.)

Основна ідея методу полягає у використанні багатоваріантних генераторів, кожен з яких виробляє свою власну послідовність псевдовипадкових чисел. Ці послідовності потім комбінуються за допомогою криптографічних хеш-функцій, які перетворюють їх у вихідну послідовність псевдовипадкових чисел.

Метод Блюма-Мікелсона використовується для створення псевдовипадкових чисел у криптографії та інших областях, де вимагається висока випадковість та стійкість до криптоаналізу. Цей метод є важливим для гарантування безпеки у багатьох криптографічних протоколах та системах.

6) Метод простих конгруентних генераторів[11] працює аналогічно до лінійних конгруентних генераторів, але використовує менші коефіцієнти та інші параметри.

7) Метод факторизації[12] використовує математичні алгоритми факторизації для генерації великих простих чисел, які можуть використовуватися для створення випадкових чисел.

8) Метод змішування включає в себе змішування кількох вже існуючих послідовностей випадкових чисел для отримання нової послідовності.

1.3 Інноваційні підходи до генерації PRN

Інноваційні підходи до генерації PRN – це нові та продуктивні методи та технології, які використовуються для створення послідовностей чисел, що відповідають концепції випадковості, навіть якщо вони засновані на детермінованих процесах. Ці інновації допомагають покращити властивості та безпеку генерованих псевдовипадкових чисел.

Зазвичай інноваційні підходи включають в себе використання новітніх технологій, таких як квантова механіка, машинне навчання, фізичні процеси або обчислювальні методи, для створення більш надійних і випадкових послідовностей чисел. Ці інновації можуть бути важливими в різних галузях, таких як криптографія, симуляції, статистика, медицина, фізика, генетика та інші, де випадкові числа використовуються для різних цілей.

Головною метою інноваційних підходів є забезпечення вищого рівня випадковості, надійності та безпеки в порівнянні з традиційними методами генерації PRN. Це може бути досягнуто шляхом використання передових технологій та наукових досліджень для створення кращих PRNG, які відповідають сучасним вимогам та потребам.

Квантова генерація випадкових чисел (QRNG) [14] – це інноваційний метод генерації випадкових чисел, який базується на принципах квантової механіки. Цей метод використовує фізичні явища на квантовому рівні для створення послідовностей чисел, які вважаються повністю випадковими і непередбачувими.

Принцип роботи QRNG:

– Квантові явища: QRNG використовує квантові явища, такі як квантовий розпад частинок, відбиття фотонів від напівпрозорих дзеркал, чи інші квантові процеси, які є повністю непередбачуваними в класичних

умовах. Квантова механіка описує, як частинки та поля ведуть себе на мікро- та нанорівні.

– Фотони та квантові стани: один з типів QRNG використовує властивості фотонів, які є квантовими частинками світла. Фотони можуть бути випущені через напівпрозору пластину, і їхні траєкторії абсолютно непередбачувані. Коли фотони взаємодіють з напівпрозорими дзеркалами, їхня доля визначається квантовими станами, що є недетермінованими.

– Вимірювання квантових станів: Другий метод QRNG полягає в вимірюванні квантових станів частинок, наприклад, спіну електронів. Відповідно до принципів невизначеності Гейзенберга, точне вимірювання квантових станів неможливе, і результати вимірювання стають випадковими.

– Повна випадковість: однією з ключових переваг QRNG є повна випадковість його результатів. Квантові явища властиві станам невизначеності, і це гарантує, що результати генерації будуть абсолютно непередбачуваними.

– Застосування в криптографії: оскільки QRNG надає повну випадковість, він ідеально підходить для створення безпечних криптографічних ключів. За допомогою QRNG можна забезпечити високу стійкість криптосистем до атак шляхом забезпечення надійності та непередбачуваності ключів.

– Обмеження та перспективи: QRNG потребує спеціалізованого обладнання, що потребує значних фінансових витрат. Проте існує активний розвиток цієї технології, і в майбутньому QRNG може стати більш доступним та ефективним.

Квантова генерація випадкових чисел є областю інтенсивних наукових досліджень, і вона має потенціал змінити підхід до безпеки і генерації випадкових чисел у багатьох сферах, включаючи криптографію, телекомунікації, медицину, фізику та інші галузі.

Генерація випадкових чисел на основі фізичних процесів [22] – це метод створення випадкових послідовностей чисел, використовуючи природні або фізичні явища. Цей метод відрізняється від PRNG, оскільки використовує непередбачувані фізичні процеси для створення випадкових даних.

Метод використовує різноманітні фізичні процеси як джерела випадковості. Наприклад:

– Тепловий шум (або шум Джонсона-Найквіста) виникає в електронних компонентах і є результатом випадкових теплових рухів електронів. Вимірювання рівня теплового шуму може використовуватись як джерело випадковості.

– Електромагнітний шум виникає внаслідок електромагнітних коливань і може бути використаний як джерело випадковості. Електромагнітний шум використовується у багатьох RNG.

– Радіоактивний розпад атомів – це непередбачуваний фізичний процес, при якому радіоактивне ядро розпадається, випромінюючи частинки або фотони. Момент розпаду є випадковим і може бути використаний для генерації випадкових чисел.

– В коливаннях кварцових кристалів є певна міра випадковості через вплив різних факторів, таких як механічні коливання та температурні зміни.

– Розсіяння фотонів один від одного – це явище виникає при взаємодії фотонів, і воно може бути використане для генерації випадкових чисел на квантових системах.

Метод генерації випадкових чисел на основі фізичних процесів відзначається надійністю та непередбачуваністю результатів. Використані фізичні процеси є надзвичайно складними та важко передбачуваними, що гарантує справжню випадковість отриманих чисел.

Зокрема, цей метод має широкий застосунок в галузі криптографії, де безпека та випадковість важливі як ніколи. Випадкові числа, що генеруються на основі фізичних процесів, дозволяють створювати надбезпечні криптографічні ключі і забезпечувати захист конфіденційної інформації.

Метод також знаходить застосування в різних галузях, включаючи статистику, наукові дослідження, моделювання, медицину та інші області. Він забезпечує надійне джерело випадкових даних для проведення аналізів та досліджень.

Важливо враховувати, що генерація випадковості на основі фізичних процесів може вимагати спеціалізованого обладнання для спостереження та реєстрації цих фізичних явищ. Це, в свою чергу, може потребувати значних фінансових витрат.

Методи генерації випадкових чисел на основі аналізу акустичних сигналів використовують звукові коливання як джерело випадковості. Наприклад, одним із підходів є використання мікрофонів, які сприймають акустичні сигнали з навколишнього середовища, такі як шум вітру, звуки природи або випадкові розмови людей. Сигнали цих джерел можуть бути відсемпльовані та піддані аналізу для отримання випадкових чисел.

Ще одним прикладом є використання апаратури, яка реєструє радіошум (шум в радіочастотному діапазоні). Радіошум є результатом різних радіочастотних явищ, таких як електромагнітні перешкоди, інтерференція та інші фактори. Цей шум може бути аналізований для створення послідовностей випадкових чисел.

Такі методи використовують акустичну або радіочастотну випадковість, що містить багато ентропії та непередбачувих змін. Вони можуть бути використані в криптографії для створення безпечних ключів, в наукових дослідженнях для моделювання складних систем, а також в інших галузях, де важлива велика ступінь випадковості. Однак для їхнього використання необхідно спеціалізоване обладнання та програмне забезпечення, що може вимагати інвестицій та технічних ресурсів.

1.4 Оцінка якості PRN

Оцінка якості PRN[13] – це процес аналізу та визначення ступеня відповідності генерованих чисел певним критеріям та вимогам щодо їхньої випадковості, незалежності, стійкості та рівномірності розподілу. Оцінка якості PRN включає в себе використання різноманітних статистичних тестів і методів для перевірки генерованих послідовностей чисел на відповідність цим критеріям.

Основні аспекти оцінки якості PRN включають:

– Рівномірність розподілу: правильні PRNG повинні генерувати числа, які розподілені рівномірно. Це означає, що кожне можливе значення має майже однакову ймовірність виникнення, і немає значущих відхилень в розподілі чисел.

У ідеально рівномірному розподілі всі значення мають однакову ймовірність появи, і це створює прямокутну або рівну гістограму генерованих чисел. Однак у практиці рівномірність може бути трохи відхиленою через обмеженість обсягу випадкової послідовності, рівень шуму та особливості алгоритму генерації.

Для оцінки рівномірності розподілу генерованих чисел використовують різноманітні статистичні тести. Один із найпростіших тестів – це тест на «частоту появи». Він рахує кількість входжень кожного значення в послідовності та порівнює їх з очікуваним значенням у рівномірному розподілі. Якщо різниця між фактичними та очікуваними частотами значною, це може свідчити про нерівномірність розподілу.

Забезпечення рівномірності розподілу є важливим для численних застосувань, де важлива висока ступінь випадковості, таких як криптографія, статистика, симуляції та інші. Рівномірність генерованих чисел допомагає уникнути біасу та гарантувати, що всі можливі значення виникають з приблизно однаковою ймовірністю.

– Незалежність: це важливий аспект оцінки, оскільки кожне число повинно бути незалежним від попередніх та наступних чисел в послідовності.

Незалежність важлива з точки зору багатьох застосувань, особливо в статистиці, симуляціях і криптографії. У статистиці, коли використовуються випадкові числа для проведення експериментів або вибірки, незалежність гарантує, що вимірювання не спотворюються систематичними помилками. В симуляціях, де важливо моделювати реальні процеси, незалежність допомагає досягти точних результатів.

У криптографії, яка використовує випадкові числа для створення криптографічних ключів, незалежність має вирішальне значення. Якщо значення генеруються залежними одне від одного, це може вразити безпеку системи, оскільки можуть з'явитися шаблони чи кореляції, які допоможуть зловмисникам розкрити ключ. Незалежність гарантує, що кожне випадкове значення надійно захищене і не може бути передбаченим на основі інших значень.

Оцінювання незалежності генерованих чисел використовує різні статистичні тести, які перевіряють, чи відповідають вони статистичним критеріям незалежності. Забезпечення незалежності генерованих чисел є важливою задачею при розробці ефективних RNG для різних застосувань.

– Статистична стійкість: оцінка стійкості PRN до різних видів статистичного аналізу та атак. Стійкість важлива для застосувань, де генеровані числа використовуються для забезпечення конфіденційності та безпеки даних, зокрема в криптографії.

Приклади статистичних аналізів та атак, якими можуть бути піддані PRN:

– Тест співвідношення перевіряє, наскільки рівномірно розподілені числа в інтервалі [0, 1]. Якщо числа виявляють якусь систематичну

відмінність від рівномірного розподілу, це може свідчити про проблеми в статистичній стійкості.

– Тест серій оцінює, наскільки незалежні числа відносно один одного. Подовжені серії однакових чисел або закономірності між числами можуть свідчити про проблеми в статистичній стійкості.

– Тест на кореляцію визначає, чи існують кореляції між числами. Наявність кореляцій може вказувати на статистичну незалежність чисел.

– Тест Колмогорова-Смірнова використовується для порівняння розподілу генерованих чисел з рівномірним розподілом. Відхилення від рівномірності може бути індикатором проблем у статистичній стійкості.

– Криптоаналітичні атаки спрямовані на розкриття або злам криптографічних ключів, часто вимагають аналізу PRN. Наприклад, атака «перебір всіх можливих ключів» може бути виконана, якщо PRN не є статистично стійким і дозволяє передбачити значення ключів.

– Тест на відповідність законам: деякі атаки можуть визначати відповідність генерованих чисел до певних законів або математичних моделей, що також може вплинути на статистичну стійкість.

– Критерії тестування: включають в себе створення статистичних тестів та критеріїв, які допомагають визначити відповідність генерованих чисел вищезазначеним критеріям, а також дублюють тести, які використовуються для статистичного аналізу та атак, яким можуть бути піддані PRN.

Висновки до розділу 1

У цьому розділі висвітлено основні концепції, теорії та методи генерації випадкових чисел. Починаючи з вступу в тему випадкових чисел, було досліджували як класичні, так і інноваційні методи генерації. Вивчення якості та оцінка створених випадкових чисел визначити їх використовуваність та застосування в різних сферах.

Цей розділ демонструє важливість випадкових чисел у сучасному світі, показавши їх широке застосування в криптографії, статистиці, моделюванні та інших областях. Розгляд класичних та передових методів генерації показує що вибір методу генерації випадкових чисел має важливе значення для точності та надійності отриманих результатів.

Оцінка якості випадкових чисел є ключовим етапом, що дозволяє визначати придатність чисел для різних завдань та додаткових вимог. Отже, цей розділ надав важливі базові знання та інструменти для подальшого вивчення, розробки та використання випадкових чисел у науковому дослідженні та практичному застосуванні.

РОЗДІЛ 2
АКУСТИЧНІ СИГНАЛИ

2.1 Огляд основних характеристик акустичних сигналів

Акустичні сигнали – це звукові коливання, які передають інформацію чи сповіщають про події. Вони грають важливу роль в комунікації та сигналізації в різних сферах життя, включаючи технологію, медицину, транспорт і багато інших.

Визначення акустичних сигналів у вигляді самостійного терміну не має конкретної дати виникнення, оскільки концепція звуку та його використання в різних сферах життя розвивалася протягом багатьох століть. Однак, з поширенням технологій і розвитком науки, акустичні сигнали стали предметом систематичного вивчення і досліджень. Вивчення акустичних сигналів стало актуальним у багатьох наукових галузях, включаючи акустику, фізику звуку, інженерію та інші. Термін «акустичні сигнали» став корисним для опису та класифікації звукових сигналів, які використовуються для передачі інформації або сповіщення в різних аплікаціях.

До основних характеристик акустичних сигналів відносять [16]:

– Амплітуда – вказує на максимальне відхилення частини повітря від його нормального стану. Велика амплітуда відповідає гучному звуку, тоді як мала амплітуда вказує на тихий звук. Амплітуда вимірюється в децибелах (дБ) та відображає рівень гучності сигналу. Велика амплітуда відповідає більшим значенням дБ, а мала – меншим.

– Частота показує, скільки разів коливається повітря за секунду під впливом звукових хвиль. Частота вимірюється в герцах (Гц). Низькі частоти відповідають низьким, глибоким звукам, а високі частоти – високим, гострим звукам. Наприклад, нота ля в музичному ладі має частоту приблизно 440 Гц.

– Тривалість визначає, скільки часу триває сигнал. Вона вимірюється у часових одиницях, таких як секунди, мілісекунди чи хвилини.

Наприклад, короткий дзвінок мобільного телефону може тривати кілька секунд, водночас як пісня або аудіокнига може тривати декілька хвилин або годин.

– Форма сигналу описує графічну репрезентацію звукових коливань залежно від часу. Вона може бути простою і містити лише одну основну частоту, як у синусоїдальному сигналі, або складною з багатьма частотами, як у складних музичних інструментах. Форма сигналу впливає на сприйняття та характер звуку.

– Режим роботи сигналу визначає, як сигнал змінюється з часом. Постійний сигнал залишається незмінним протягом усього часу. Імпульсний сигнал може мати амплітуду, частоту чи фазу, що змінюються з часом. Наприклад, амплітудно-модульовані сигнали змінюють амплітуду залежно від інформації, яка передається.

– Напрямок поширення сигналу вказує на те, чи він поширюється у певному напрямку чи поширюється на всі сторони. Спрямовані сигнали, наприклад, використовуються для спрямованого звукового сповіщення, тоді як неспрямовані сигнали, які поширюється на всі сторони, можуть бути використані для створення загального акустичного поля.

– Джерело сигналу показує, звідки сигнал походить. Це може бути активне джерело, що генерує звук, наприклад, музичний інструмент або мікрофон, або пасивне, що просто випромінює звукові коливання під впливом іншого джерела. Джерело визначає, як сигнал створюється і де починає шлях передачі.

– Фаза сигналу вказує на положення звукових коливань відносно початку коливань. Вона вимірюється у градусах або радіанах і вказує на часовий зсув між коливаннями. Фазова інформація може бути важливою для стереофонічного аудіо та обробки сигналів.

– Спектральний склад сигналу вказує на те, які частоти складових звукових хвиль входять в сигнал. Аналіз спектра дозволяє визначити, які

фреквенції присутні в сигналі та їхні амплітуди. Це важливо для розрізнення різних звукових джерел і для обробки сигналів.

– Динамічний діапазон сигналу вказує на різницю між максимальною та мінімальною амплітудою сигналу. Великий динамічний діапазон означає, що сигнал має широкий спектр гучності від тихого до гучного, що важливо для вірної передачі звуку в аудіозаписах та музиці.

2.2 Опис методів запису та обробки акустичних даних

Для роботи з акустичними сигналами використовуються методи запису та обробки акустичних даних, що включають в себе процес фіксації звукових сигналів та їх подальшу обробку для аналізу, зберігання, передачі або відтворення. Ці методи широко використовуються в таких галузях, як акустика, музика, аудіозапис, телекомунікації, медицина та багато інших.

До методів запису акустичних даних відносять[19]:

– Мікрофони: використовуються для перетворення звуку в електричні сигнали. Вони різняться за типом (наприклад, конденсаторні, динамічні, п'єзоелектричні) та спрямованістю (наприклад, кардіоїдні, однозвіркові, омнідирекційні), що дозволяє вибирати мікрофон, який найкраще підходить для конкретних завдань.

– Аудіоінтерфейси: для запису акустичних даних на комп'ютер часто використовуються аудіоінтерфейси, які підключають мікрофони і інші джерела звуку до комп'ютера. Це може бути зовнішній або внутрішній пристрій.

– Звукові карти: звукові карти в комп'ютерах дозволяють записувати та відтворювати аудіо. Вони мають аналогові та цифрові входи і виходи для підключення звукового обладнання.

– Гідрофони: гідрофони використовуються для запису звуків у водному середовищі, таких як звуки морських істот або звук судноплавства.

– Гідерофони: гідерофони застосовуються для запису звуків в гірському середовищі, наприклад, в гірських річках та потоках.

Методи обробки акустичних даних [7]:

1) Попередній запис та фільтрація: перш ніж акустичні дані будуть оброблятися, їх можуть фільтрувати для видалення небажаних шумів та попередньої обробки.

Попередній запис та фільтрація – це важлива частина обробки акустичних даних, спрямована на попередню обробку сигналу перед подальшим аналізом чи збереженням. Цей процес дозволяє покращити якість акустичних даних та зменшити вплив шумів. Для даного етапу можна визначити такий порядок:

– Попередній запис – це етап обробки акустичних даних, який включає в себе підготовку сигналу для подальшого аналізу. Основні завдання попереднього запису включають в себе:

– Збір даних включає запис аудіосигналу з джерела (наприклад, мікрофону) та конвертацію акустичних коливань у цифровий формат. Зазвичай, цифровий аудіосигнал представляється у вигляді дискретних відліків на основі аналогового сигналу.

– Семплювання полягає у виборці точок в часі для цифрового представлення акустичного сигналу. Важливо вибирати досить високу частоту семплювання, щоб зберегти деталі сигналу.

– Квантування полягає у визначенні кількості бітів для кодування кожного відліку. Це впливає на точність та роздільну здатність аудіоданих.

– Фільтрація акустичних даних використовується для виділення або приглушення певних частот або компонентів сигналу. Основні види фільтрації включають в себе: фільтрацію низької частоти (призначений для приглушення високочастотних компонентів сигналу, що може використовуватися для видалення шуму, який зазвичай має високу частоту); фільтрацію високої частоти (призначений для приглушення низькочастотних

компонентів сигналу, що може бути використаний для виділення високочастотних подій чи для видалення низькочастотного шуму); фільтрація смуги частот (дозволяє виділити певну смугу частот та приглушити інші, використовується, коли потрібно виділити конкретний діапазон частот); фільтрація ручного вбору (використовується для приглушення конкретних частот чи шумових компонентів, які можуть бути видалені з сигналу).

Фільтрація допомагає покращити якість сигналу та зменшити вплив шуму на подальший аналіз або відтворення акустичних даних.

2) Амплітудна та часова обробка акустичних даних – це важливі аспекти обробки звукового сигналу, спрямовані на аналіз та маніпуляцію амплітудою та часовою характеристикою звуку. Ці процеси можуть використовуватися для покращення звучання аудіо, виділення особливих акустичних подій, а також для аналізу звуку у різних застосуваннях. Ось більше інформації про амплітудну та часову обробку:

Амплітудна обробка стосується маніпуляцій амплітудою аудіосигналу, тобто змінення його гучності або інтенсивності. Основні аспекти амплітудної обробки включають:

– Збільшення амплітуди включає в себе підвищення гучності аудіосигналу. Він може бути використаний для підсилення слабкого звуку або збільшення виразності звуку.

– Зменшення амплітуди слугує для зниження гучності сигналу. Це може бути важливо для видалення шумів чи контролю гучності сигналу.

– Компресія дозволяє обмежити різницю між найвищою та найнижчою амплітудою сигналу. Вона часто використовується для зниження динамічного діапазону сигналу.

– Змішування амплітуд двох або більше аудіосигналів може створювати новий сигнал, який поєднує їхні амплітуди.

Часова обробка акустичних даних стосується аналізу та маніпуляції часовими характеристиками сигналу. Основні аспекти часової обробки включають:

– Затримка аудіосигналу в часі дозволяє зсувати або синхронізувати різні компоненти сигналу.

– Зрізання сигналу в часі використовується для видалення непотрібних початкових чи кінцевих частин сигналу.

– Збільшення чи зменшення тривалості дозволяють змінювати тривалість сигналу без зміни його частотної характеристики.

– Аналіз часових пат тернів включає в себе вимірювання і аналіз змін у часі, такі як часові інтервали між подіями.

Амплітудна та часова обробка дозволяють створювати та маніпулювати аудіосигналами для різних цілей, включаючи відтворення, обробку для аудіопродукції, аналіз аудіоданих та багато інших застосувань. Спектральний аналіз: для вивчення спектрального складу акустичних сигналів використовуються методи, які розкривають наявність та інтенсивність різних частот.

3) Аналіз часової-частотної області – це важливий метод обробки акустичних даних, який дозволяє одночасно аналізувати зміни амплітуди сигналу в часі та частотній області. Цей метод дозволяє вивчати спектральну структуру сигналу та його динаміку в часі.

Основні методи часово-частотного аналізу аудіосигналу включають в себе:

– Швидке перетворення Фур'є (FFT): Цей метод використовується для перетворення сигналу з часової області в частотну область. Він розкриває, які частоти присутні у сигналі і як їх інтенсивність змінюється в часі.

– Спектрограма – це візуальне представлення аудіосигналу в часово-частотній області. Вона відображає інтенсивність різних частот у

залежності від часу та дозволяє визначити, які частоти переважають у певних моментах часу.

– Вейвлет-аналіз дозволяє використовувати вейвлет-функції для аналізу як часових, так і частотних характеристик сигналу одночасно.

– Часово-частотні траєкторії: Цей метод дозволяє відстежувати рухи часових та частотних характеристик сигналу в часі. Він корисний для аналізу змін динаміки сигналу.

Аналіз часової-частотної області знаходить застосування в різних галузях, включаючи музичний аналіз (визначення акордів, визначення ритму), розпізнавання мови, визначення звуку оточуючого середовища (наприклад, для роботів або систем розпізнавання інтерфейсів), аудіо- та звукозапису, медичних дослідженнях та багатьох інших сфер.

Аналіз часової-частотної області допомагає отримати глибше розуміння аудіосигналів та використовувати їх для різних завдань аналізу та обробки.

4) Розпізнавання мови та розпізнавання звуку: використовуються алгоритми для визначення, що саме було сказано в аудіозаписі або який саме звук був записаний (наприклад, розпізнавання мелодії).

Розпізнавання мови [18]– це процес перетворення вимовленого слова або тексту в електронний текстовий формат. Для розпізнавання мови з аудіосигналу виділяються характеристики, такі як Mel-Frequency Cepstral Coefficients (MFCC), які репрезентують акустичні властивості мовлення. Далі для розпізнавання мови використовуються мовні моделі, які включають в себе ймовірності входження слів в послідовності та граматичні правила. Вхідний акустичний сигнал порівнюється зі створеною моделлю, і обчислюється ймовірність того, що вхідний сигнал відповідає конкретному слову чи фразі. На основі порівняння і ймовірності відбувається розпізнавання мовлення, і результат подається у вигляді тексту або команди. Розпізнавання мови широко використовується в системах голосового

управління, голосових асистентах, системах розпізнавання мови на телефонних лініях та багатьох інших застосуваннях.

Розпізнавання звуку [17] – це процес ідентифікації та класифікації аудіосигналів, які не обов'язково є мовними. Аналогічно до розпізнавання мови, акустичні сигнали підлягають виділенню ознак, таких як MFCC або спектральні характеристики. Моделі для розпізнавання конкретних звуків або шумів навчаються на великому обсязі аудіоданих. Після навчання моделей вхідний звук порівнюється з попередньо навченими моделями, і сигнал класифікується на основі найбільш ймовірної категорії.

Розпізнавання звуку може бути використано для визначення різних аудіосигналів, таких як музичні композиції, звуки тварин, шуми машин та інші. Воно також застосовується в системах безпеки, відеоспостереженнях, автоматичних системах визначення подій та багатьох інших сферах.

5) Аудіообробка для зменшення шуму: шумопідгнічуючі методи використовуються для відокремлення корисного сигналу від шуму. Наприклад, фільтрація шуму – використовуються фільтри для виділення корисного сигналу та приглушення шуму. Це можуть бути лінійні фільтри, такі як фільтри низьких та високих частот, або нелінійні фільтри, такі як медіанні фільтри. Спектральний аналіз – метод полягає в аналізі спектрального складу аудіосигналу та виділенні шумових компонентів. Зазвичай використовують методи FFT (швидкого перетворення Фур'є) для цього. Адаптивне фільтрування – у цьому методі використовують адаптивні фільтри, які автоматично налаштовуються для виділення шуму та підганяються під конкретні умови. Спресування та кодування звуку – може зменшити розмір файлу та виключити деякий шум, але цей метод також може призвести до втрати якості аудіо. Машинне навчання і нейромережі – застосування нейромереж та машинного навчання дозволяє створювати моделі, які можуть визначити та видалити шум з аудіосигналів на основі навчання на великій кількості даних.

6) Компресія та кодування[6] –це методи стиснення аудіоданих з метою зменшення їх обсягу без втрати важливої інформації. Це дозволяє зберігати аудіосигнали при меншому розмірі файлів або передавати їх через мережу з меншими витратами на пропускну здатність.

Компресія аудіосигналу включає в себе дві основні стратегії – без втрати та з втратами. Компресія без втрат дозволяє зберегти аудіодані без втрати якості. Популярні формати для безвтратної компресії включають FLAC та ALAC. Вони здатні стиснути аудіодані без втрати якості і відновити оригінальний сигнал точно таким, яким він був перед компресією. При компресії з втратами деяка інформація втрачається під час стиснення, але створюється компактна версія сигналу, яка залишається прийнятною для відтворення. Популярні формати для стиснення аудіосигналів з втратами включають MP3, AAC та OGG. Ці формати використовують різні методи, такі як видалення високочастотних компонентів або апроксимація амплітуди для досягнення стиснення.

Кодування аудіосигналу – це процес представлення аудіоданих у вигляді бінарного коду для зберігання або передачі. До методів кодування відносяться: PCM (Pulse Code Modulation) – це безвтратний формат, який кодує аудіосигнал, використовується у форматах WAV та AIFF; ADPCM (Adaptive Differential Pulse Code Modulation) – використовує стиснення для кодування різниці між сусідніми відліками, що дозволяє зменшити обсяг даних; DPCM (Differential Pulse Code Modulation) – це метод, який кодує різницю між поточним та попереднім значеннями сигналу, що також сприяє стисненню; кодування з втратами – як вже згадувалося, формати, такі як MP3, використовують кодування з втратами для стиснення аудіосигналу – вони враховують психоакустичні особливості сприйняття звуку людиною та видаляють менш важливі аудіодеталі; AAC (Advanced Audio Coding) – ефективний формат з втратами, який зазвичай надає кращу якість звуку в порівнянні з MP3 при однаковому бітрейті.

Компресія та кодування аудіосигналів грають важливу роль в сучасних системах зберігання та передачі аудіоданих, допомагаючи зменшити розмір файлів та витрати на пропускну здатність, при цьому забезпечуючи прийнятну якість аудіо.

7) Обробка ефектів та обробка для аудіо продукції [23] – це важливий аспект в створенні та підготовці аудіоматеріалів для різних медіа-проектів, включаючи музику, фільми, аудіокниги та інше.

Обробка ефектів – це застосування різних аудіо-ефектів для модифікації та покращення звуку. Деякі поширені аудіо-ефекти включають:

– Реверберація (Reverb) додає просторовий відгук до аудіозапису, імітуючи відлуння в приміщенні

– Exo (Delay) створює відлуння аудіосигналу, що повторюється з певною затримкою.

– Хорус (Chorus) додає ефект «хору» шляхом копіювання та затримки аудіосигналу, створюючи враження масового звучання.

– Фленжер (Flanger) створює артикульований звук за допомогою комбінації затримки та змінного фазового зсуву.

– Дисторшн (Distortion) змінює амплітуду сигналу для створення гітарних або інших рок-звуків.

Обробка для аудіопродукції включає в себе широкий спектр дій, спрямованих на створення якісних аудіоматеріалів для конкретного медіа-проекту. Наприклад:

– Мікшування (Mixing) аудіодоріжок, включаючи баланс гучності, панорамування та об'єднання різних аудіоджерел в одну композицію.

– Мастеринг (Mastering) включає в себе налаштування обробки, компресії, еквалізації та інших аудіо-ефектів для забезпечення однакової якості та гучності всієї аудіодоріжки альбому чи іншого проєкту.

– Звуковий дизайн (Sound Design) – створення та редагування звукових ефектів, які використовуються у фільмах, відеоіграх та інших медіапроектах.

– Синхронізація (Synchronization) Забезпечує точну синхронізацію аудіодоріжок з відеоабо іншими мультимедійними елементами.

– Обробка шуму (Noise Reduction) видаляє нежиттєвий шум та інтерференцію з аудіозаписів.

2.3 Математичні та фізичні моделі акустичних сигналів

Математичні та фізичні моделі акустичних сигналів [20]– це теоретичні конструкції, які використовуються для пояснення та прогнозування поширення звукових хвиль у різних середовищах. Математичні моделі базуються на різноманітних математичних підходах та рівняннях, які описують хвильові процеси. Фізичні моделі використовують фізичні закони та властивості середовищ для розуміння та симуляції поширення звуку через конкретні матеріали або середовища.

Математичні моделі акустичних сигналів використовують математичні рівняння для опису поширення звуку у середовищі. Ці моделі можуть бути аналітичними, ґрунтуючись на рівняннях хвильової теорії, або чисельними, використовуючи обчислювальні методи, такі як метод скінченних елементів чи метод скінчених різниць для симуляції поширення акустичних хвиль.

Лінійний часовий ряд у термінах обробки сигналів та статистики представляє собою послідовність числових значень, отриманих в різні моменти часу або в однакових інтервалах. Це може бути будь-що, від температурних даних, фінансових показників, до сигналів від приладів, таких як акустичні сигнали чи електричні коливання.

Перетворення Фур'є є одним з найважливіших математичних інструментів у аналізі сигналів. Воно дозволяє розкласти сигнал або функцію від часу на її складові частоти. За допомогою цього перетворення можна

розглядати сигнал у термінах його частотної складової, що розкриває частотні компоненти, які утворюють цей сигнал.

Це важливо для аналізу сигналів у різних контекстах: у технічних аплікаціях, наприклад, у обробці звуку, визначенні частот природних явищ у науці, у медицині для аналізу біомедичних сигналів та в багатьох інших областях, де необхідно розуміти структуру та складові сигналу у частотному діапазоні.

Хвильовий аналіз переважно є математичним інструментом, що використовується для аналізу хвильових явищ у різних системах. Це складна математична методика, що дозволяє розкласти сигнал чи хвилю на її складові частоти за допомогою перетворення Фур'є або інших математичних інструментів. Хоча він ґрунтується на фізичних принципах хвиль та їх поширення, саме перетворення та аналіз є в основному математичними концепціями, які допомагають розкрити частотні характеристики сигналу. Цей аналіз дозволяє визначати складові частоти сигналу та їх характеристики, але як сам метод аналізу, він відноситься переважно до математичних інструментів для дослідження сигналів.

Спектральний аналіз – це метод аналізу сигналів, який використовується для вивчення їхньої структури у частотному діапазоні. В основі спектрального аналізу лежить розклад сигналу на його складові частоти чи частотний склад, який відображає, які конкретні частоти складають сигнал.

Цей аналіз дає змогу перетворити сигнал із часового представлення у представлення у частотній області. Одним із основних інструментів для спектрального аналізу є перетворення Фур'є, яке дозволяє розкласти сигнал на його частотні складові.

Спектральний аналіз використовується в різних галузях, включаючи сейсмологію, акустику, радіофізику, обробку сигналів, медицину та інші області. Цей аналіз дозволяє отримати важливі відомості про структуру

сигналів, виявляти частоти, домінуючі у сигналі, та здійснювати подальший аналіз сигналів у частотному спектрі.

Спектральний аналіз вважається математичним методом аналізу сигналів. Він використовує математичні інструменти для розкладання сигналу на його складові частоти. Хоча його застосовують у фізичних науках та інженерії для дослідження фізичних явищ, сам спектральний аналіз базується на математичних принципах та інструментах, зокрема на теорії сигналів і систем, математичному розкладанні Фур'є та інших методах.

Цей аналіз використовується для вивчення характеристик сигналів у частотному діапазоні та для отримання інформації про складові частоти сигналу. Його математична природа дозволяє розглядати сигнал з точки зору його частотного складу та виявляти частотні характеристики сигналу.

Фізичні моделі акустичних сигналів використовують фізичні закони та властивості середовищ для розуміння та симуляції поширення звуку через конкретні матеріали або середовища. Вони враховують параметри середовища, такі як щільність, пружність та акустична імпеданса, а також акустичні властивості матеріалів, такі як поглинання, відбиття та розсіювання звукових хвиль.

Акустичні моделі, що використовують фізичні закони поширення звуку у конкретних матеріалах або середовищах називаються моделями поширення звуку у середовищі. Вони враховують властивості середовища, як щільність та пружність, для передбачення, як звук розповсюджується через ці середовища.

Моделі акустичних властивостей матеріалів досліджують, як різні матеріали взаємодіють з звуковими хвилями. Вони враховують, чи поглинають, відбивають чи розсіюють звукові хвилі матеріали, що їх оточують.

Моделі акустичних систем охоплюють дослідження акустичних систем, які включають мікрофони, динаміки, фільтри та інші акустичні

компоненти. Вона враховує явища, пов'язані зі збором, передачею, обробкою та відтворенням звукових сигналів.

2.4 Застосування акустичного аналізу

Акустичний аналіз[4] – це комплексний підхід до дослідження звукових сигналів, який включає в себе вивчення їхньої структури, характеристик та взаємодії з навколишнім середовищем. Цей метод використовує різні техніки обробки сигналів та аналізу, щоб отримати більше інформації зі звукових даних.

Використання акустичного аналізу розповсюджується на безліч галузей. В музичній сфері він використовується для обробки, зміни та покращення музичних творів, а також для створення нових звукових ефектів.

У медицині акустичний аналіз застосовується для вивчення та діагностики різних станів, таких як аудіологічні порушення або для аналізу внутрішніх органів через їх звукові властивості.

Також він використовується для безпеки та в технічних аспектах, виявляючи важливість у виявленні аномальних звукових сигналів для прогнозування небезпечних ситуацій або для контролю якості у виробництві.

Акустичний аналіз надає можливості для глибшого розуміння звукових сигналів та використання цієї інформації для різноманітних цілей у різних галузях.

В акустичному аналізі виділення випадкових складових сигналів відіграє важливу роль у різних сферах, включаючи розпізнавання мови, аналіз звуку та обробку сигналів. Однією з основних задач в акустичному аналізі є розрізнення корисних сигналів від шуму або небажаних складових, що можуть перешкоджати адекватній обробці аудіоінформації.

Виділення випадкових складових сигналів полягає у виявленні, вилученні та ізоляції неочікуваних чи непередбачуваних елементів в аудіосигналах. Це може включати розпізнавання та виділення голосу від

фонового шуму, розуміння окремих звукових подій у складних аудіозаписах, аналіз акустичних сигналів у мовних системах та інше.

Одним із методів для виділення випадкових складових є використання спеціалізованих алгоритмів обробки сигналів, таких як фільтри, спектральний аналіз, машинне навчання тощо. Наприклад, алгоритми фільтрації можуть бути застосовані для приглушення або вилучення певних частот або шумів з аудіозапису.

Технології глибокого навчання та штучний інтелект також широко використовуються для розв'язання завдань виділення випадкових складових сигналів у аудіо. Шляхом навчання моделей на великих обсягах даних можна розробити системи, які автоматично розрізняють та видаляють інформацію з шумом чи іншими небажаними аудіоелементами.

У контексті розпізнавання мови, наприклад, виділення випадкових складових може допомогти в розумінні мовлення в умовах шумного оточення або при існуванні багатьох спікерів. В процесі розпізнавання мови системи аналізують аудіозапис та спробують виокремити акустичні особливості кожного спікера (наприклад, тембр голосу, швидкість мовлення, вимова звуків тощо). Це допомагає відрізняти різних спікерів один від одного, що може бути корисним у випадках, коли необхідно визначити, хто саме говорить на аудіозаписі або в аудіовідтворенні. Це допомагає покращити точність систем автоматичного перекладу мови та інших мовних технологій.

Такі техніки знаходять застосування не лише у сферах розпізнавання мови, а й у звуковому мистецтві, музичній продукції та інших галузях, де важливо виділити чи вилучити певні аудіоелементи для створення бажаного звукового ефекту або обробки аудіоінформації.

Акустичний аналіз може бути використаний для генерації випадкових чисел через використання акустичних фізичних процесів[5], що ґрунтуються на шумі та випадкових коливаннях у середовищі.

Шум у звукових сигналах, наприклад, може бути використаний для генерації випадкових чисел. Аналізуючи акустичний шум або властивості аудіосигналів, можна використовувати цю випадковість для створення послідовностей випадкових чисел.

Один з методів полягає у використанні фізичних пристроїв, які перетворюють акустичні сигнали на електричні сигнали. Наприклад, мікрофони можуть фіксувати оточуючий звуковий шум, а цей шум подається на аналізуючі електронні схеми. Після цього, на основі певних властивостей аудіосигналу, можуть бути згенеровані випадкові числа, використовуючи, наприклад, алгоритми криптографічного перетворення.

Принцип використання акустичних сигналів для генерації випадкових чисел полягає в тому, що навіть у звучанні, яке може здаватися регулярним, існує певний рівень непередбачуваності та випадковості, який може бути використаний для створення послідовностей випадкових чисел.

Проте, варто зауважити, що цей підхід може бути вразливим до зовнішніх впливів або специфічних умов довкілля, що можуть змінювати чи впливати на отримані випадкові числа.

Висновки до розділу 2

У цьому розділі було розглянуто широкий спектр аспектів, пов'язаних з акустичним аналізом та його потенційними застосуваннями. Починаючи з аналізу акустичних сигналів, було описано їх основні характеристики та методи запису та обробки. Дослідивши математичні та фізичні моделі акустичних сигналів, включаючи лінійний часовий ряд, перетворення Фур'є, хвильовий аналіз та спектральний аналіз, було отримано глибше розуміння природи звукових даних.

Окрім цього, було розглянуто застосування акустичного аналізу та вивчено виділення випадкових складових сигналів, оцінку параметрів акустичних сигналів та можливість використання акустичного аналізу для генерації випадкових чисел. Дослідження акустичних сигналів було необхідним для подальшої роботи з темою, оскільки воно є ключовим у контексті генерації випадкових чисел. Акустичний аналіз відкриває можливості для виявлення випадкових компонентів у сигналах та оцінки їх параметрів. Ці дані можуть послужити основою для генерації випадкових чисел, використовуючи особливості акустичних сигналів та їхніх властивостей. Таким чином, аналіз звукових даних не лише поглибив розуміння природи звукових сигналів, але й надав важливі вихідні дані для подальших досліджень в генерації випадкових чисел.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ ГЕНЕРАЦІЇ ВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ АНАЛІЗУ АКУСТИЧНИХ СИГНАЛІВ

3.1 Вибір інструментів розробки

Перед початком реалізації системи генерації випадкових чисел на основі аудіосигналів було проведено аналіз декількох можливих варіантів мов програмування та середовищ розробки, таких як:

– MATLAB/Octave: ці мови часто використовуються в області сигнальної обробки та аналізу звуку. Вони надають зручні інструменти для роботи з аудіоданими.

– Java: якщо важлива міжплатформенність або інтеграція з іншими системами, Java може бути обрана завдяки своїй універсальності.

– C++: якщо вимагається висока швидкість обробки даних, особливо при роботі з великими обсягами аудіоданих.

– R: для випадків, коли акцент робиться на статистичному аналізі та моделюванні.

– JavaScript: якщо планується веб-застосунок або інтеграція з веб-серверами для обробки аудіоданих в реальному часі.

Python був обраний як мова програмування для даної магістерської роботи через його універсальніть, широке застосування у галузі наукових досліджень та аналізу даних. Висока читабельність синтаксису Python сприяє легкості розробки та розумінню коду, що важливо у великих та складних проектах. Мова має велику та активну спільноту розробників, яка підтримує різноманітні бібліотеки та інструменти для вирішення різних завдань.

Щодо середовища розробки, PyCharm було обрано через свою потужність та зручність для роботи з Python. PyCharm надає широкий набір інструментів, включаючи автоматичне доповнення коду, інтегровану систему відлагодження, аналіз якості коду, підтримку віртуальних середовищ та

багато інших функцій. Його інтерфейс та функціонал дозволяють розробникові зосередитися на самому дослідженні та аналізі, забезпечуючи при цьому продуктивну робочу обстановку.

У роботі використовуються спеціалізовані бібліотеки для обробки аудіоданих. Наприклад, бібліотека Librosa надає функції для аналізу аудіосигналів, включаючи витягування спектрограм, обчислення середніх амплітуд та інше. Numpy та Scipy використовуються для числових обчислень та статистичного аналізу, дозволяючи отримувати точні та надійні результати. Matplotlib використовується для візуалізації отриманих даних, що допомагає зрозуміти результати та зробити висновки.

Цей комплексний підхід до вибору мови та інструментів розробки створює потужну та зручну робочу обстановку для вирішення завдань, пов'язаних з аналізом та обробкою аудіоданих.

3.2 Розробка алгоритму та архітектури системи

Перед початком реалізації системи генерації випадкових чисел на основі аналізу аудіосигналів було поставлено завдання, вирішення яких визначало основні етапи роботи системи. Основними етапами роботи системи генерації є робота зі звуком, його аналіз та генерація випадкових чисел на основі даного аналізу. Відповідно до поставлених задач, було розроблено алгоритм, що передбачає ефективну робота з аудіоданими та здатність адаптуватися до різних параметрів аналізу.

Загальний алгоритм роботи системи продемонстровано у блок-схемі (Рис.3.1), що відповідає реалізації та демонструє кожен із етапів, необхідний для коректної роботи системи та відповідності очікуваним результатам.

Рис.3.1. Блок-схема алгоритму системи генерації

На початковому етапі створення системи генерації випадкових чисел проводиться обробка звукової інформації. Цей етап включає в себе використання спеціалізованої бібліотеки для запису та обробки аудіоданих. Крім того, здійснюється збереження отриманої аудіодоріжки з метою подальшого порівняння та аналізу результатів.

Наступний етап розробки передбачає введення параметрів системи користувачем. На цьому етапі визначається діапазон значень, кількість

генерованих чисел та обирається тип параметра для генерації. Це робить систему більш гнучкою та налаштовуваною під потреби користувача.

Третім етапом є генерація чисел на основі аналізу аудіосигналу. Для цього використовуються різні методи, залежно від обраних параметрів. Серед них – врахування середнього амплітудного значення, спектральний аналіз, випадковий зсув та інші.

Важливим елементом архітектури системи є стандартизація послідовності згенерованих чисел. Цей процес дозволяє привести дані до єдиної форми для подальшого порівняння та аналізу.

Завершальним етапом є виведення результатів у вигляді графіків та гістограм. Це включає побудову графіків для згенерованих послідовностей чисел та їх кількостей, а також для стандартизованих послідовностей. Додатково, забезпечено збереження отриманих послідовностей у файли та обробка можливих помилок для забезпечення надійності системи.

3.3 Реалізація алгоритму системи

У програмі для системи генерації випадкових чисел на основі аналізу аудіосигналів було використано наступні бібліотеки:

– «sounddevice» –ця бібліотека використовується для запису аудіосигналу. Вона забезпечує інтерфейс для роботи з аудіо-програмним обладнанням комп'ютера, дозволяючи записувати звук з мікрофону.

– «librosa»– використовується для аналізу аудіосигналів. Вона надає ряд функцій для обробки та аналізу аудіоданих, таких як обрізка та екстракція характеристик сигналу.

– «matplotlib.pyplot» – використовується для візуалізації результатів. При реалізації алгоритмуїї було застосовано для побудови гістограм для відображення розподілу згенерованих чисел.

– «numpy»– використовується для ефективної роботи з числовими даними в Python. Безпосередньо в системі генерації використовується для

обробки та маніпуляції числовими даними, наприклад, для обчислення статистичних характеристик аудіосигналу чи роботи зі згенерованими числами.

– «soundfile»– ця бібліотека дозволяє зберігати аудіодані у файловій системі. У реалізації програми використовується для збереження аудіо-данихпісля їх обробки та аналізу.

– «sys» – використовується для взаємодії із системою. У системі генерації використовуєтьсядля виходу з програми у випадку помилки або некоректного запису аудіосигналу.

– «random»: – для генерації випадкових чисел. У програмі використовується для різноманітних генераційних завдань, наприклад, для випадкового вибору стовпців у спектральному аналізі.

Основна функція програми передбачає покроковий виклик функцій, що реалізують основні завдання описані в алгоритмі системи:

```
if __name__ == "__main__":
    audio_data = record_audio()
    audio_trimmed = analyze_audio(audio_data)
    user_input = get_user_input()
    generated_sequence,  element_counts =
generate_numbers(audio_trimmed, user_input)
    standardized_sequence_result, standardized_element_counts =
standardize_sequence(generated_sequence, user_input)
    display_element_counts(element_counts, user_input,
is_normalized=False)
    display_element_counts(standardized_element_counts,
user_input, is_normalized=True)
```

Функція «record_audio» відповідає за запис звуку:

```
def record_audio(duration=5, fs=44100):
    print("Розпочато запис аудіо...")
    audio_data = sd.rec(int(duration * fs), samplerate=fs,
channels=1)
```

```
    sd.wait()
    print("Запис завершено.")
    return audio_data.flatten()
```

Дана функція передбачає параметри «duration=5, fs=44100», де «duration» визначає тривалість запису в секундах (за замовчуванням – 5 секунд), а «fs» визначає частоту дискретизації (за замовчуванням – 44100 Гц)

Використовуючи бібліотеку sounddevice, наступна команда розпочинає запис аудіосигналу.

```
audio_data = sd.rec(int(duration * fs), samplerate=fs,
channels=1)
```

Параметр «int(duration * fs)» визначає кількість відгуків для запису, враховуючи тривалість та частоту дискретизації. «channels=1» вказує, що запис проводиться для одного аудіоканалу.

Команда «sd.wait()» очікує завершення запису, гарантуючи, що функція поверне аудіо-дані тільки після завершення запису.

Функція «return audio_data.flatten()» повертає аудіо-дані у вигляді одновимірного масиву, використовуючи метод «.flatten()». Це зручно для подальшого обробки та аналізу.

Отже, функція «record_audio» використовує бібліотеку «sounddevice» для запису аудіосигналу, приймаючи враховані параметри, та повертає одновимірний масив, представляючи записаний аудіо. Також функція має відповідні виведення повідомлень для інформування користувача про стан запису звуку.

Наступною функцією є «analyze_audio», що включає обробку, аналіз та збереження отриманих аудіо-даних на основі яких вже відбуватиметься подальша генерація чисел.

```
def analyze_audio(audio_data):
    if np.all(audio_data == 0):
        print("Звук записано некоректно. Повторіть спробу.")
```

```
        input("Натисніть Enter для виходу з програми...")
        sys.exit()
    else:
        audio_trimmed = librosa.effects.trim(audio_data,
frame_length=2048, hop_length=512)[0]
        sf.write('analyzed_audio.wav', audio_trimmed, 44100)
        print("Записані аудіо-дані збережено у файл
'analyzed_audio.wav'.")
        return audio_trimmed
```

Для виклику функції використовуються вже записані аудіодані «audio_data», що потребують аналізу та обробки. Спочатку відбувається перевірка, чи звук був записаний коректно (відповідність, що дані, отримані при запису звуку, не рівні нулю). У випадку некоректного запису, функція виводить повідомлення про помилку та виходить з програми.

Якщо звук був записаний правильно, функція використовує бібліотеку «librosa» для обрізання аудіоданих, використовуючи параметри «frame_length=2048 та hop_length=512». «frame_length» (довжина кадру) вказує, скільки аудіоданих включається у кожен кадр. Велика довжина кадру дозволяє отримати більше часової інформації, але при цьому обчислення можуть бути витратнішими. Вказане значення 2048 обране для того, щоб забезпечити досить дрібні кадри для отримання деталізованої інформації про аудіосигнал. «hop_length» (крок зсуву) вказує, на скільки аудіоданих зсувається вікно під час переходу від одного кадру до наступного. Значення 512 обране для забезпечення перекривання між кадрами та отримання більшої частотної інформації.

Отримані обрізані аудіо-дані зберігаються у файл з назвою «analyzed_audio.wav» із частотою дискретизації 44100 Гц.

Після збереження аудіо-даних, функція повертає обрізану аудіодоріжку для подальшого використання в програмі.

Наступною функцією для подальшої роботи є «get_user_input», що взаємодіє з користувачем для отримання параметрів, необхідних для генерації випадкових чисел на основі аудіосигналу. Розглянемо окремо отримання кожного із параметрів, що використовуються у системі генерації випадкових чисел на основі аналізу аудіосигналів.

Першим елементом, без якого неможлива генерація випадкових чисел, є межі діапазону, в якому відбувається генерація:

```
def get_user_input():
    lower, upper = None, None
    while lower is None or upper is None or lower > upper or
lower == upper:
        try:
            lower = int(input("Введіть нижню межу діапазону: "))
            upper = int(input("Введіть верхню межу діапазону:
"))
            if lower > upper:
                print("Нижня межа діапазону повинна бути меншою
за верхню.")
                lower, upper = None, None
            elif lower == upper:
                print("Межі діапазону не можуть бути
однаковими.")
                lower, upper = None, None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть цілі
числа.")
            lower, upper = None, None
```

«lower» (нижня межа) та «upper» (верхня межа) визначають діапазон чисел. Користувач вводить нижню та верхню межі, а функція перевіряє їх на коректність (наприклад, чи нижня межа не більше верхньої чи є введені межі числом).

Далі необхідно отримати інформацію про те, скільки чисел потребує користувач у свою послідовність:

```
    num_generated = None
    while num_generated is None:
        try:
            num_generated = int(input("Введіть бажану кількість генерованих чисел: "))
            if num_generated <= 0:
                print("Недопустиме значення. Кількість має бути більше за 0.")
                num_generated = None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть ціле додатнє число.")
            num_generated = None
```

«num_generated» визначає кількість чисел, які будуть згенеровані. Користувач вводить це значення, а функція перевіряє його на допустимість (більше за 0 та чи є введене значення числом).

Наступним етапом є вибір користувачем параметру, за яким відбуватиметься генерація.

```
audio_param_type = None
    while audio_param_type is None:
        try:
            print("")
            audio_param_type = int(input("""
Параметри для генерації:
    1 - Середнє амплітудне значення аудіосигналу
    2 - Спектральний аналіз
    3 - Середня амплітуда з використанням передекспозиції
    4 - Середня амплітуда з використанням передекспозиції та випадковий зсув
    5 - MFCC
Оберіть параметр: """))
```

```
            if audio_param_type <= 0 or audio_param_type > 5:
                print("Недопустиме значення. Тип параметру має
бути в межах від 1 до 5.")
                audio_param_type = None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть ціле
додатнє число.")
            audio_param_type = None
```

«audio_param_type» визначає, який аудіопараметр використовувати для генерації випадкових чисел. Користувач обирає один з варіантів, і функція перевіряє його на коректність. Користувач має можливість вказати такі параметри, як середнє амплітудне значення аудіосигналу, спектральний аналіз, середня амплітуда, середня амплітуда та випадковий зсув, а також характеристику аудіосигналу MFCC.

Ця функція гарантує, що користувач вводить правильні значення, щоб вони можна було використовувати для генерації випадкових чисел на основі аудіосигналу.

Повертає функція, відповідно, межі діапазону (нижню та верхню), кількість генерованих чисел та тип параметру генерації:

```
    return lower, upper, num_generated, audio_param_type
```

Наступна функція «generate_numbers» використовує аудіосигнал для створення послідовності випадкових чисел, враховуючи обрані параметри користувача. Суть роботи функції полягає у взаємодії з аудіоданими та виборі певного аспекту сигналу для генерації чисел. Користувач обирає один з п'яти параметрів («audio_param_type»), які визначають, як саме аналізувати аудіосигнал. Отримана послідовність чисел записується у файл «generated_sequence.txt».

До основних кроків функції належать:

1) Ініціалізація параметрів: розпаковка введених користувачем параметрів, таких як нижня та верхня межі, кількість генерованих чисел і обраний тип параметру.

2) Аналіз аудіосигналу: використання бібліотеки librosa для виконання різних видів аналізу аудіосигналу, таких як спектральний аналіз, вимірювання середньої амплітуди та отримання коефіцієнтів MFCC.

3) Генерація чисел: залежно від обраного типу параметру (audio_param_type), генеруються числа на основі різних характеристик аудіосигналу, таких як середнє амплітудне значення, спектральний аналіз, середня амплітуда тощо.

4) Отримання інформації для гістограм: під час генерації чисел використовується окремий словник для збереження кількості повторень кожного елементу для подальшого їх використання при побудові гістограм.

5) Запис результатів: отримана послідовність чисел "generated_sequence.txt" записується у текстовий файл.

Функція дозволяє використовувати аудіосигнал як джерело випадкових чисел, де кожен обраний параметр впливає на спосіб взаємодії з аудіоданими та генерації чисел.

Перший крок передбачає розпакування даних, що були отримані з функцій «get_user_input» та «analyze_audio»:

```
def generate_numbers(audio_data, user_input):
    lower, upper, num_generated, audio_param_type = user_input
```

Для подальшої роботи при обраному аудіо-параметрі, необхідно створити ряд змінних, в яких зберігатимуться дані та використати певні функції, щоб не викликати їх при кожному проходженні циклу генерації для пришвидшення роботи системи.

Пустими елементами, в яких зберігатиметься подальша інформація є масив чисел «generated_sequence» в якому зберігатиметься генерована

послідовність, а також словник «element_counts», що використовується для збереження кількості повторень кожного елементу, який надалі стане основою для побудови гістограм генерованих послідовностей, що дозволить провести аналіз отриманих чисел та візуально оцінити отриманий розподіл навіть при генерації великої кількості значень.

Одним з параметрів, за якими відбуватиметься генерація, є середнє амплітудне значення аудіосигналу. Формула для визначення середнього значення:

$$mean = \frac{\sum_{i=1}^{N} x_i}{N}, \quad (3.1)$$

де x_i – значення аудіосигналу, N – кількість відгуків (зразків) в сигналі.

Обрана мова програмування спрощує отримання даного середнього значення та представлено функцією «np.mean», що використовує отримані аудіо-дані для аналізу:

```
mean_value = np.mean(audio_data)
```

Для отримання наступних значень було використано спеціалізовану бібліотеку «librosa», що дозволяє детально проаналізувати записані дані та виокремити з них параметри, що використовуватимуться при генерації послідовностей чисел.

Функція для обчислення спектрограми:

```
spec = librosa.feature.melspectrogram(y=audio_data, 
sr=44100, n_fft=2048, hop_length=512
```

При виклику функції, окрім аудіо-даних використано:

– sr=44100: частота дискретизації аудіосигналу (44.1 кГц).

– n_fft=2048: розмір вікна для FFT.

– hop_length=512: зміщення між послідовними кадрами у спектрограмі.

Наступна функція використовується для обчислення середнього амплітудного значення аудіосигналу з використанням передекспозиції:

```
amplitude = np.mean(np.abs(librosa.effects.preemphasis(audio_data)))
```

Де «librosa.effects.preemphasis» застосовує ефект передекспозиції до аудіосигналу. Цей ефект спрямований на виділення важливих компонентів сигналу, зменшення шумів та підвищення розпізнаваності сигналу. «np.abs» використовується для обчислення абсолютних значень кожного елементу масиву. Після застосування «librosa.effects.preemphasis», масив містить значення, де деякі можуть бути від'ємними. Взяття абсолютних значень дозволяє отримати модуль кожного елементу, тобто видалити від'ємний знак. «np.mean» обчислює середнє арифметичне значень масиву. В даному випадку, це використовується для обчислення середньої амплітуди аудіосигналу після застосування передекспозиції.

Наступна функція передбачає отримання коефіцієнтів MFCC:

```
mfcc = librosa.feature.mfcc(y=audio_data, sr=44100, n_mfcc=13)
```

Функція librosa.feature.mfcc використовує алгоритм MFCC для отримання репрезентації аудіосигналу. Основні кроки виконання цієї функції:

1) Переведення аудіосигналу в спектрограму: аудіосигнал розбивається на короткі часові вікна. Для кожного вікна обчислюється перетворення Фур'є, щоб отримати спектральний зміст.

2) Отримання енергії на різних частотах: використовуючи спектрограму, обчислюється енергія на різних частотах.

3) Створення мел-фільтрів: застосовується набір фільтрів на основі шкали Мел, які відображають спосіб, яким людське вухо сприймає звуки на різних частотах.

4) Отримання логарифмів вагованих енергій

5) Обчислення дискретного косинусного перетворення

6) Застосованнядискретного косинусного перетворення до логарифмів вагованих енергій для отримання коефіцієнтів MFCC.

При виклику функції використовуються аудіо-дані, частота дискретизації аудіосигналу та кількість коефіцієнтів MFCC. Кількість коефіцієнтів («n_mfcc=13») є стандартним вибором в багатьох застосунках обробки мови та аудіосигналів. Кількість 13 є традиційною і зазвичай вважається оптимальною для представлення основних музичних характеристик або аудіосигналів взагалі.

Основний цикл генерації випадкових чисел (Рис.3.2.) включає в себе розгалуження, в якому відповідно до обраного типу аудіо-параметру відбувається генерація (від 1 до 5), далі генероване число додається до послідовності «generated_sequence», після чого генероване число перевіряється на знаходження у словнику element_counts та збільшує кількість даного числа на один.

Перший параметр відповідає за генерацію випадкового числа на основі середнього амплітудного значення аудіосигналу:

```
if audio_param_type == 1:
    generated_number = int(np.random.normal(mean_value,
num_generated / 3))
    while True:
        generated_number = max(min(generated_number, upper + 1),
lower - 1)
        if generated_number != lower - 1 and generated_number !=
upper + 1:
            break
        generated_number = int(np.random.normal(mean_value,
num_generated / 3))
```

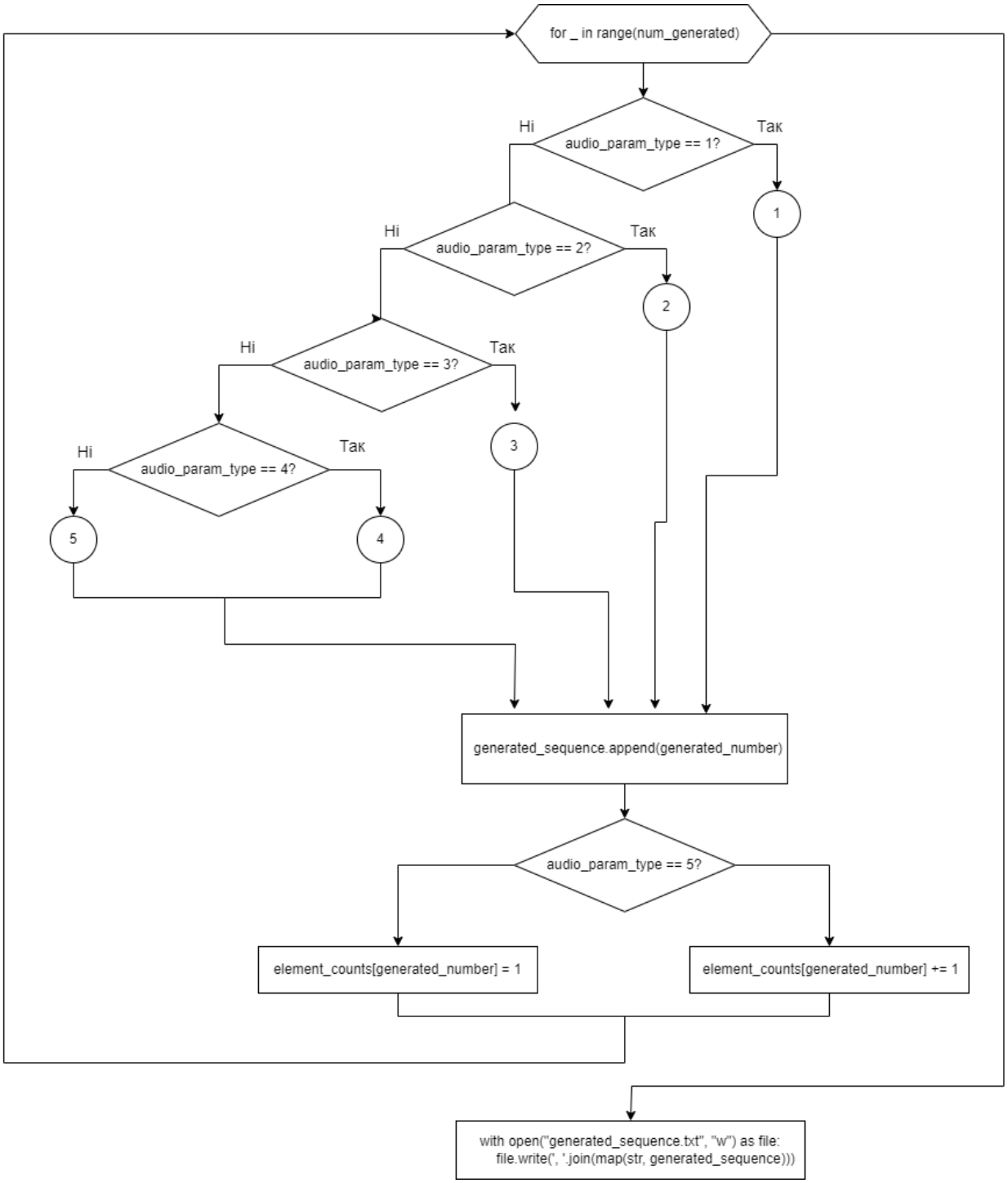

Рис.3.2. Основний цикл генерації випадкових чисел

Перший рядок

```
generated_number = int(np.random.normal(mean_value,
num_generated / 3))
```

здійснює генерацію випадкового числа на основі нормального розподілу, використовуючи середнє значення («mean_value») та третину

кількості генерованих чисел (num_generated / 3). Вибір саме третини кількості розраховано дослідним шляхом при побудові гістограм при підрахунку кількості повторень кожного числа в послідовності.

Далі запускається безкінечний цикл для перевірки та виправлення генерованих чисел, адже, при нормалізації чисел до вказаних меж, буде завелика кількість елементів, що відповідають lower (нижній межі) та upper (верхній межі), через особливість роботи функції нормалізації до бажаних користувачем меж.

```
while True:
    generated_number = max(min(generated_number, upper + 1), 
lower - 1)
    if generated_number != lower - 1 and generated_number != 
upper + 1:
        break
    generated_number = int(np.random.normal(mean_value, 
num_generated / 3))
```

Перший рядок циклу перевіряє генероване число в межах діапазону [lower-1, upper+1], тобто чи не виходить генероване число за межі діапазону (менше «lower-1» та більше «upper+1»). Якщо виходить, воно заміщується межею. При програмуванні від меж віднімається\додається число один задля збереження у подальшій генерації можливості генерованому числу дорівнювати одній з меж, що були задані користувачем.

Далі відбувається перевірка, чи не є генероване число рівне «lower-1» та «upper+1», в даному випадку цикл завершується, якщо ні – то відбувається повторна генерація випадкового числа до тих пір, поки генероване число не стане знаходитись в межах [lower, upper].

Другим параметром, за яким відбуватиметься генерація послідовності є генерація на основі спектрального аналізу, частину якого вже було проведено на початку роботи функції (обчислення спектрограми«spec»).

```
elif audio_param_type == 2:
```

Спочатку необхідно здійснити вибірку 10 випадкових стовпців із спектрограми. «spec.shape[1]» вказує на кількість стовпців у спектрограмі, і «random.sample» використовується для вибору 10 випадкових індексів цих стовпців:

```
column_index = random.sample(range(spec.shape[1]), 10)
```

Далі отримуємо значення вибраних стовпців зі спектрограми – «spec[:, column_index]» витягує всі рядки для вибраних стовпців:

```
selected_column = spec[:, column_index]
```

Залишилось нормалізувати значення стовпчика до діапазону [lower, upper]:

```
generated_number = int( (np.max(selected_column) -
np.min(selected_column)) / (np.max(spec) - np.min(spec)) *
(upper - lower) + lower)
```

Вираз використовує максимальне та мінімальне значення вибраного стовпчика та спектрограми для коректної нормалізації. Згідно з вказаним діапазоном [lower, upper], визначається, яке значення має бути згенероване.

Наступний аудіо-параметр відповідає за генерацію чисел на основі середнього амплітудного значення аудіосигналу з використанням передекспозиції:

```
elif audio_param_type == 3:
          generated = lower + np.random.rand() * (upper - lower)
* amplitude
```

Даний вираз створює випадкове число в межах [0, 1) та масштабує його за допомогою середньої амплітуди. Здійснюється генерація числа у межах [lower, upper].

Далі визначається скільки розрядів використовувати для кожного числа, створення числа в межах вказаного діапазону та забезпечення, що значення знаходились в межах [lower, upper]:

```
    scale_factor = 10 ** 5
    scaled_numbers = lower + (generated * scale_factor) % (upper - lower)
    generated_number = np.round(np.clip(scaled_numbers, lower, upper)).astype(int)
```

Кількість розрядів, що використовуються для перевірки було обрано при обробці результатів генерації даного аудіо-параметра до використання розширення розрядності числа, так як генеровані числа знаходились в межах [0, 1) незалежно від того, які межі були вказані користувачем та представляли собою десяткові числа вигляду «0,0xx» з 16-ма знаками після коми.

«scaled_numbers» створює нове число, яке враховує «scale_factor» та діапазон [lower, upper]. Використовується оператор залишку «%», щоб періодично отримувати значення в межах вказаного діапазону. Така періодичність може виникнути, коли залишок ділення повторюється, і числа «повертаються» до початкового значення у вказаному діапазоні.

При отриманні «generated_number» забезпечується, що значення знаходяться в межах [lower, upper], відбувається округлення числа та перетворення його у ціле.

Четвертим параметром на вибір для генерації випадкового числа є на основісередньої амплітуди аудіосигналу з використанням передекспозиції та випадкового зсуву:

```
elif audio_param_type == 4:
    shift = np.random.uniform(lower, upper)
    scaled_number = int(np.round((shift / amplitude) * np.random.uniform(0, amplitude)))
    generated_number = min(upper, max(lower, scaled_number))
```

Функція «np.random.uniform()» генерує випадкове число з рівномірним розподілом від «lower» (включно) до «upper» (не включно). Іншими словами, випадкове число буде більше або рівне lower та менше upper. Це випадкове число «shift» потім використовується для генерації випадкового зсуву, який впливає на значення в межах заданого діапазону.

Далі значення зсуву масштабується на основі середньої амплітуди «amplitude» та випадкового зсуву «shift». Отримане значення «scaled_number» конвертується у ціле число.

Останнім етапом забезпечується, що значення знаходиться в межах [lower, upper]. Це досягається за допомогою функцій «min()» та «max()», які обмежують значення відповідно до верхньої та нижньої межі.

П’ятим параметром, що використовується для генерації випадкового числа є характеристика аудіосигналу MFCC.

```
else:
    normalized_mfcc = (mfcc - mfcc.min()) / (mfcc.max() -
mfcc.min())
    mfcc_vector = normalized_mfcc.flatten()
    generated_numbers = np.random.choice(mfcc_vector,
num_generated)
    generated_numbers_scaled = np.round(lower +
generated_numbers * (upper - lower)).astype(int)
     unique_numbers = np.unique(generated_numbers_scaled)
     selected_numbers = np.random.choice(unique_numbers,
num_generated)
    generated_number = np.random.choice(selected_numbers)
```

Цей фрагмент коду виконує наступні кроки:

1. Нормалізація MFCC до діапазону [0, 1]: Обчислюється нормалізована версія MFCC, щоб забезпечити, що всі значення знаходяться в межах від 0 до 1.

2. Перетворення MFCC у вектор: Отримані нормалізовані MFCC перетворюються в одновимірний вектор, використовуючи функцію «flatten()». Це необхідно для того, щоб наступним кроком було зручно брати випадковий набір чисел.

3. Генерація чисел від 0 до 1 на основі MFCC: З використанням функції np.random.choice() генерується масив чисел від 0 до 1 на основі отриманого вектора MFCC. Кількість згенерованих чисел визначається параметром num_generated.

4. Операції над генерованими числам для масштабування до вказаного діапазону: Згенеровані числа, які спочатку знаходяться в межах [0, 1], масштабуються і трансформуються, щоб потрапити в заданий користувачем діапазон [lower, upper].

5. Отримання унікальних чисел та випадковий вибір: Спочатку отримуються унікальні числа з попередньо згенерованого набору. Після цього випадковим чином обирається одне число з цього набору, яке і стане згенерованим числом для даного етапу.

Останній етап з отримання унікальних чисел є обов’язковим, адже без нього, при перевірці гістограми розподілу генерованої послідовності, отримані числа не достатньо різноманітні для забезпечення коректної генерації. Хоч надалі отримана послідовність і проходить через функцію нормалізації «standardize_sequence», для користувачів, що хочуть використати не стандартизовану послідовність – на етапі генерації також варто урізноманітнити отримані числа.

Далі при проходженні кожної ітерації числу та отриманні генерованого числа «generated_number» воно додається до послідовності, для подальшої обробки та зберігання:

```
generated_sequence.append(generated_number)
```

На цьому ж етапі використовується словник «element_counts» для відстеження кількості повторень кожного генерованого числа в процесі

виконання функції. Відстеження кількості на даному етапі необхідно для того, щоб у подальшому, при виклику функції побудови гістограм, не програмувати додатковий цикл з перебором елементів генерованої послідовності.

```
if generated_number in element_counts:
    element_counts[generated_number] += 1
else:
    element_counts[generated_number] = 1
```

Якщо «generated_number» вже є ключем у словнику «element_counts», то інкрементується значення, що відповідає цьому ключу. Таким чином, ведеться облік кількості зустрічей даного числа. Якщо «generated_number» не є ключем у словнику, то відбувається додавання нового ключа, прив'язаного до значення 1. Це вказує на те, що генероване число зустрічається вперше.

Після завершення циклу та генерування послідовності чисел та словника з підрахованою кількістю значень кожного числа в послідовності отримана послідовність записується у файл, а також повертаються значення генерованої послідовності та словника для подальшої роботи з ними:

```
with open("generated_sequence.txt", "w") as file:
    file.write(', '.join(map(str, generated_sequence)))
return generated_sequence, element_counts
```

При запису відкривається файл з ім'ям «generated_sequence.txt» у режимі запису ("w"). Всі числа у «generated_sequence» перетворюються у рядок і записуються у файл, розділені комами (', '). Файл автоматично закривається після завершення блоку коду через використання контекстного менеджера (with).

Після отримання генерованої послідовності буде використано функцію «standardize_sequence», що призначена для генерації стандартизованої послідовності чисел на основі послідовності генерованої на основі аналізу аудуосигналу.

Для роботи функції передаємо в неї генеровану послідовність чисел, яку потрібно стандартизувати та параметри користувача, включаючи нижню та верхню межі діапазону та кількість генерованих чисел. Тип аудіо-параметра не використовується, тому його опускаємо.

```
def standardize_sequence(sequence, user_input):
    lower, upper, num_generated, _ = user_input
```

Першим етапом, необхідним для стандартизації генерованої послідовності, є встановлення послідовності в діапазоні [0, 1]. Обчислюються мінімальне та максимальне значення вхідної послідовності (min_value, max_value), які надалі використовуються при обробці:

```
min_value = np.min(sequence)
max_value = np.max(sequence)
standardized_sequence = (sequence - min_value) / (max_value - min_value)
```

Наступним кроком обчислюється функція кумулятивної ймовірності для стандартизованої послідовності «standardized_sequence»:

```
    cumulative_prob = np.cumsum(standardized_sequence) / np.sum(standardized_sequence)
```

Ця лінія коду використовує бібліотеку NumPy для обчислення кумулятивної ймовірності (кумулятивного розподілу) на основі стандартизованої послідовності чисел.

«np.cumsum» – це функція NumPy, яка обчислює кумулятивну суму для елементів у масиві. Кожен елемент результуючого масиву є сумою всіх елементів до відповідного індексу у вихідному масиві «standardized_sequence». Іншими словами, для елемента на позиції «i» в результуючому масиві він обчислює суму:

$$cumulative_{sum[i]} = \sum_{k=0}^{i} standardized\_sequence[k], \qquad (3.2)$$

де:

$cumulative_{sum[i]}$ – кумулятивна сума до індексу i

$standardized\_sequence[k]$ – k-тий елемент стандартизованої послідовності,

сума від $k = 0$ до i включно.

Розділяємо кожен елемент у кумулятивній сумі на загальну суму всіх елементів у вихідному масиві «standardized_sequence» («np.sum(standardized_sequence)»). Це призводить до отримання кумулятивної ймовірності для кожного елемента, яка визначає, яка частина від загальної ймовірності припадає на або перед кожним елементом послідовності.

Наступним етапом генеруємо випадкові числа для нової послідовності:

```
random_values = np.random.rand(num_generated)
```

Далі застосовується функція кумулятивної ймовірності для отримання нових значень відповідно до розподілу вхідної послідовності, що була генерована основі аналізу аудіо-сигналу:

```
standardized_sequence = np.round(np.interp(random_values, cumulative_prob, np.linspace(lower, upper, len(sequence)))).astype(int)
```

Цей рядок коду виконує генерацію нової стандартизованої послідовності, використовуючи функцію кумулятивної ймовірності. Розглянемо його детальніше:

– np.linspace(lower, upper, len(sequence)): генерує лінійну рівномірно розподілену послідовність значень від lower до upper з кількістю значень, що дорівнює довжині вхідної послідовності.

– np.interp(random_values, cumulative_prob, ...): використовує функцію лінійної інтерполяції для знаходження відповідних значень для кожного випадкового числа в random_values на основі кумулятивної

ймовірності cumulative_prob. Це перетворення дозволяє взяти випадкові значення і замінити їх на нові значення, які розподілені згідно кумулятивної ймовірності.

– np.round(...): округлює отримані значення до найближчого цілого числа.

– .astype(int): конвертує значення до цілого типу.

Отже, результатом цього виразу є нова стандартизована послідовність, згенерована на основі функції кумулятивної ймовірності та випадкових значень.

Аналогічно для використання у побудові графіків розподілу генерованої послідовності, необхідно створити словник для збереження кількості повторень елементів стандартизованої послідовності для візуальної оцінки якості генерованої послідовності та порівняння послідовності з стандартизацією та без неї:

```
element_counts = {}
    for number in standardized_sequence:
        if number in element_counts:
            element_counts[number] += 1
        else:
            element_counts[number] = 1
```

Далі стандартизована послідовність зберігається у текстовому файлі «standardized_sequence.txt»:

```
with open("standardized_sequence.txt", "w") as file:
    file.write(', '.join(map(str, standardized_sequence)))
```

По завершенню функції «standardize_sequence» повертаються стандартизована послідовність та словник, що містить кількість повторень для кожного елементу.

Остання функція, що використовується у програмі є «display_element_counts», що відповідає за візуалізацію розподілу значень та

їхніх кількостей у генерованій послідовності. Дана функція викликається двічі з різними вхідними параметрами – окремо для генерованої послідовності на основі аналізу аудіо-сигналу та стандартизованої послідовності:

```
def display_element_counts(element_counts,user_input,
is_normalized):
    lower, upper, num_generated, audio_param_type = user_input
```

Вхідний параметр функції «is_normalized» при виклику функції приймає значення False при побудові графіка послідовності без нормалізації та True – з нормалізацією. Це знадобиться у подальшому для відображенні коректних повідомлень з інформацією для користувача.

Також при виклику функції використовується словник значень (для генерованої та стандартизованої послідовності відповідно), а також користувацькі значення, необхідні для побудови графіків, а також для виведення інформації у вікно.

Для аудіо-параметра, що використовувався для вибору користувачем принципу за яким відбувалась генерація чисел, проводимо додаткову обробку:

```
if audio_param_type == 1:
    param_type = "Середнє амплітудне значення аудіосигналу "
elif audio_param_type == 2:
    param_type = "Спектральний аналіз"
elif audio_param_type == 3:
    param_type = "Середня амплітуда з використанням
передекспозиції"
elif audio_param_type == 4:
    param_type = "Середня амплітуда з використанням
передекспозиції та випадковий зсув"
else:
    param_type = "Характеристика аудіосигналу MFCC"
```

Для побудови графіків необхідно розділити отриманий у функцію словник на два списки: значення та кількості повторень:

```
elements = list(element_counts.keys())
counts = list(element_counts.values())
```

Після цього використовуємо бібліотеку «Matplotlib» для відображення графіків послідовностей:

```
plt.figure(figsize=(10, 6))
plt.bar(elements, counts, color='b', alpha=0.7)
plt.xlabel('Значення')
plt.ylabel('Кількість повторень')
plt.gcf().canvas.manager.window.wm_title(f'Межі:
[{lower},{upper}], Кількість чисел: {num_generated}, Тип
параметра: {param_type}')
```

Створюється нова фігура для графіка з розмірами 10 на 6 дюймів. Розміри фігури вказуються в дюймах і впливають на загальний розмір графіку.

Виклик «plt.bar» генерує гістограму, де «elements» – значення по осі X, а «counts» – їх кількість по осі Y. Параметр «color='b'» встановлює колір графіку на синій, а «alpha=0.7» задає прозорість.

Виклик «plt.xlabel» та «plt.ylabel» додає підпис по осі X та Y відповідно до значень та кількостей повторень елементів послідовності.

Виклик «plt.gcf().canvas.manager.window.wm_title(...)» встановлює заголовок вікна графіку. Значення заголовка формується за допомогою форматованого рядка (f-string), де «lower», «upper», «num_generated», і «param_type» замінюються на відповідні значення.

Цей фрагмент коду встановлює основні параметри графіку та забезпечує його правильне відображення з вказаними підписами та заголовком.

Заключною частиною є встановлення назви графіка залежно від того чи є послідовність стандартизованою та виведення графіка на екран:

```
if is_normalized:
    plt.title('Генерована послідовність після стандартизації')
    plt.show()
else:
    plt.title('Генерована послідовність')
    plt.show(block=False)
    plt.pause(1)
```

При розробці програми виникла проблема із одночасним виведенням графіків з нормалізацією та без: при звичайному виклику функції «plt.show()» програма очікувала закриття першого графіку і тільки після цього будувала других графік. Для вирішення цієї проблеми перший виклик відбувається для генерованої послідовності без стандартизації з додатковим параметром «plt.show(block=False)».

При вирішенні проблеми сформувалась ще одна – одночасна побудова двох графіків послідовностей при великій кількості генерованих елементів та широкого діапазону значень призводить до зависань програми після візуалізації першого графіку генерованої послідовності без стандартизації. Для вирішення цієї проблеми було використано «plt.pause(1)» для створення паузи перед викликом наступної функції побудови графіку, що дозволяє візуалізовувати одночасно послідовності уникаючи зависань програми для слабких пристроїв.

Як результат, функція «display_element_counts» візуалізує розподіли значень та їхніх кількостей у генерованій послідовності без стандартизації та зі стандартизацію. Відкриті вікна з графіками мають відповідні підписи, що свідчать про обрані користувачам параметри діапазону, кількості значень, обраного аудіо-параметра для генерації випадкових чисел, а також відповідні назви послідовностей та їх розподіл.

3.4 Тестування системи та огляд генерованих послідовностей

Для представлення результатів роботи основним пунктом, від якого залежить огляд результатів є вхідні параметри, що передає користувач: звук, що записується програмою, нижня та верхня межа діапазону та бажана кількість елементів. Демонстрація результатів буде представлена у декількох варіантах для різних аудіо-параметрів без використання стандартизації та зі стандартизацією.

На вибір меж діапазону та кількості елементів впливає багато факторів, такі як особливості застосування, фізичні обмеження аудіосигналу та точність роботи системи. Більша кількість елементів може допомогти збільшити статистичну значущість демонстрованих експериментів. З ростом кількості елементів також зростає час генерації та обробки даних.

Для встановлення меж діапазону при генерації на основі аудіосигналу в форматі 8-біт можна було б обрати [0:255], але, так як в розробленій системі підключена бібліотека Soundfile (sf) та використовується 16-бітне ціле представлення (bit depth), можна згенерувати більш широкий діапазон значень при роботі з вищою роздільною здатністю, наприклад, [0:1000]. Кількість елементів встановимо 10 000, адже більше кількість елементів допоможе збільшити статистичну значущість експериментів.

При запуску програми автоматично відбувається запис звуку з певною затримкою. Після чого відбувається обробка звуку із його подальшим записом у файл «analyzed_audio.wav», про що виводиться відповідне повідомлення (Рис.3.3).

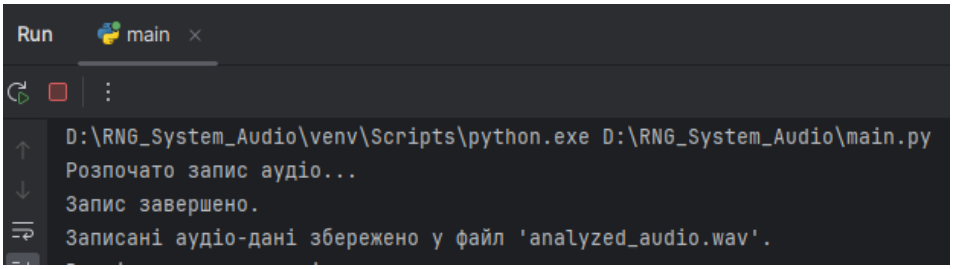

Рис.3.3 Запуск програми та запис звуку

Наступним кроком програма потребує від користувача введення даних, необхідних для генерації послідовності. Програма аналізує отримані від користувача дані та виводить відповідні повідомлення у випадку отримання некоректних даних (Рис.3.4): введення верхньої межі менше нижньої чи при рівних значеннях діапазону, введення кількості елементів менше нуля та ін.

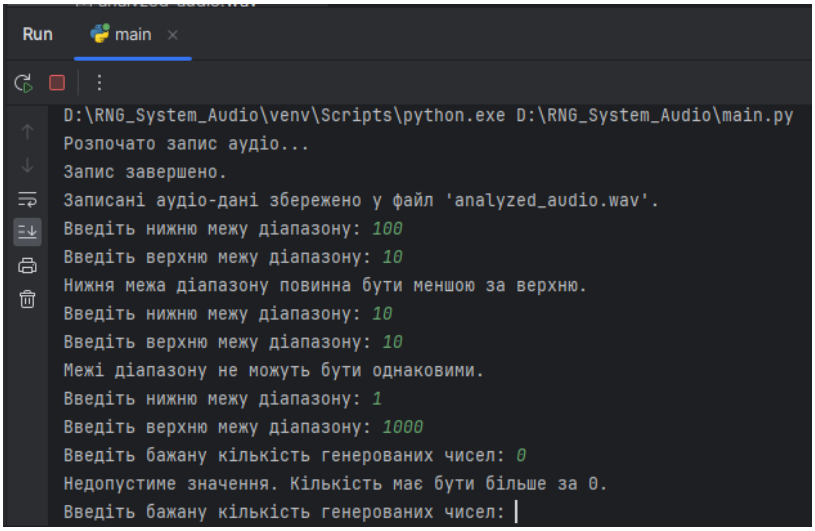

Рис.3.4 Перевірка коректності введених даних

Далі користувач має ввести аудіо-параметр відповідно до якого відбуватиметься генерація випадкових чисел (Рис.3.4). Параметр також перевіряється на коректність вводу.

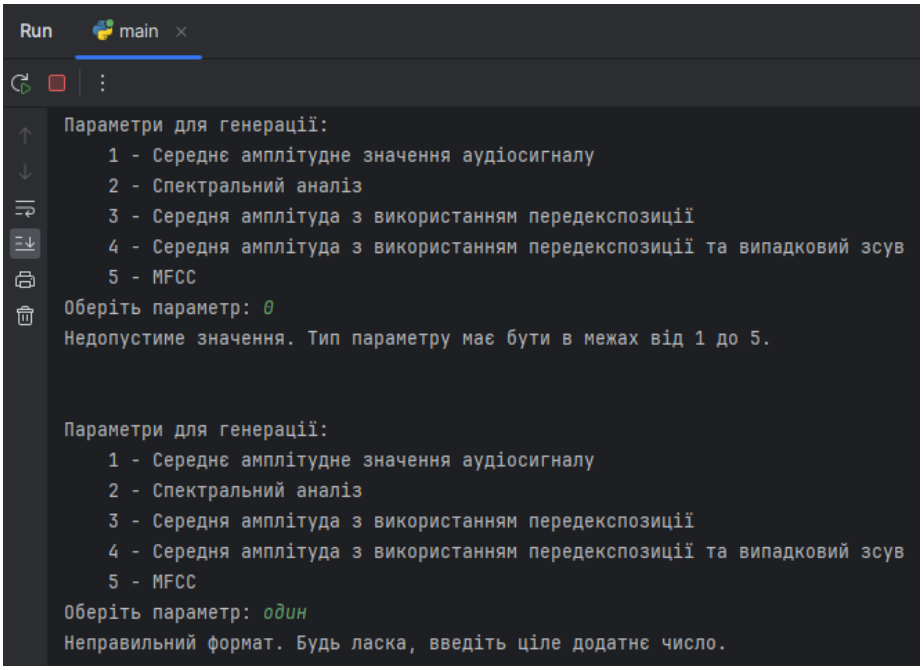

Рис.3.5 Перевірка введення аудіо-параметра

Як було встановлено вище, використаємо діапазон [1:1000] та кількість елементів 10 000 та послідовно перевіримо роботу та генеровану послідовність залежно від вибору аудіо-параметру.

1) Середнє амплітудне значення аудіосигналу

Для встановлених параметрів генерованої послідовності при виконанні програмавідкриває два вікна (Рис 3.6-3.7) із демонстрацією вказаних користувачем параметрів (назва вікна містить значення меж, кількості елементів та параметру), відповідною назвою («Генерована послідовність» для послідовності до стандартизації та «Генерована послідовність після стандартизації»). При демонстрації наступних встановлених параметрів буде відображено виключно графік для генерованих послідовностей без вказання параметрів для більш детального розгляду саме розподілу без акцентування на інтерфейсі користувача.

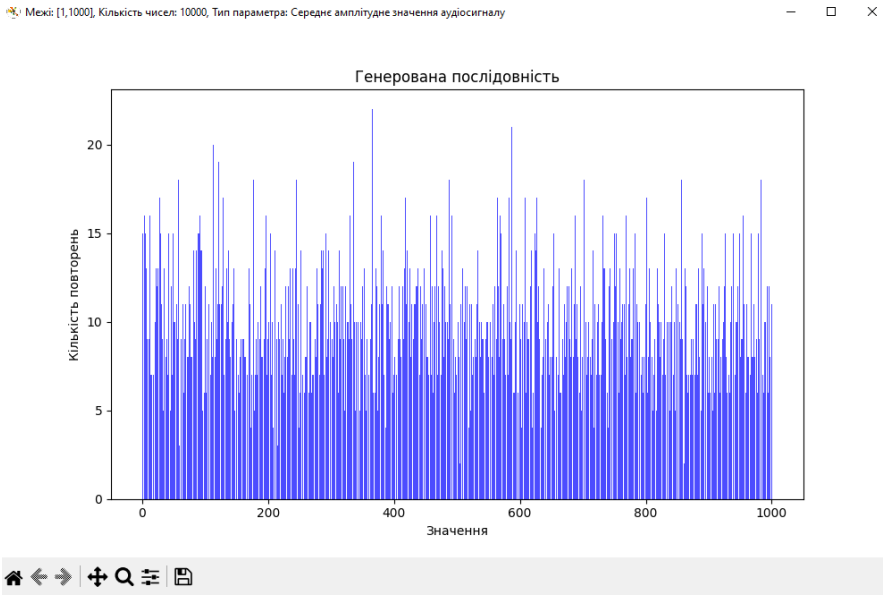

Рис. 3.6 Генерована послідовність на основі середнього амплітудного значення аудіосигналу

В даному випадку генерована послідовність на має критичної необхідності у використанні стандартизації та дозволяє користувачу використовувати «оригінальну» генеровану послідовність.

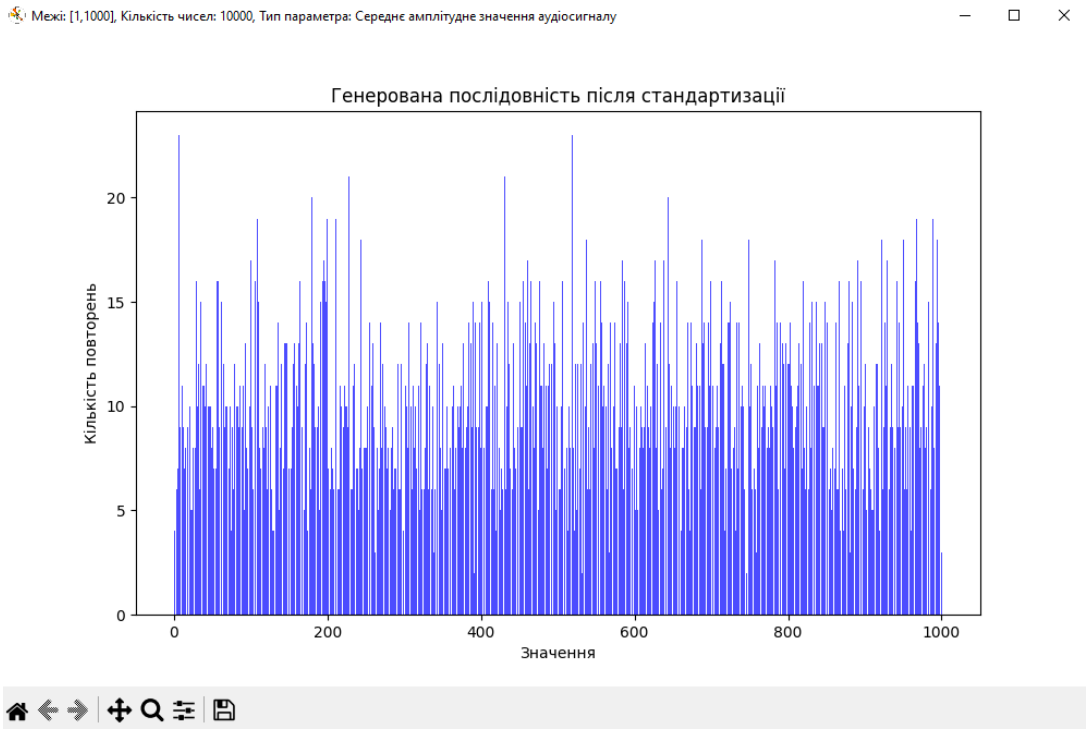

Рис. 3.7 Генерована послідовність на основі середнього амплітудного значення аудіосигналу після стандартизації

Програма очікує закриття вікон перед тим як завершити свою роботу, а також записує отримані послідовності у відповідні файли «generated_sequence.txt» та «standardized_sequence.txt». Файли представляють собою документ із набором числових значень розділених комою (Рис. 3.8).

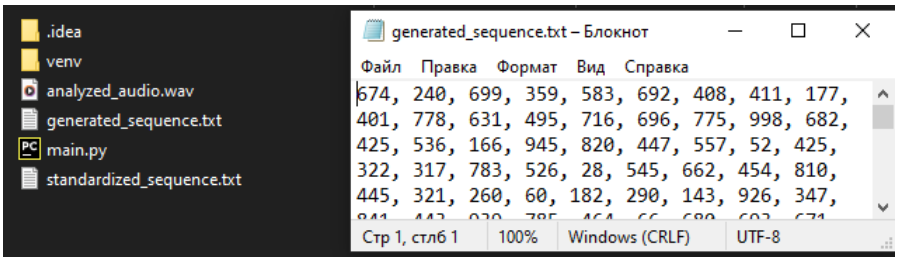

Рис.3.8 Робоча папка системи та файл генерованої послідовності

2) Спектральний аналіз

При виборі аналогічних меж та кількості елементів спектральний аналіз демонструє менш прийнятний розподіл елементів через особливості характеристик аудіосигналу (Рис.3.9).

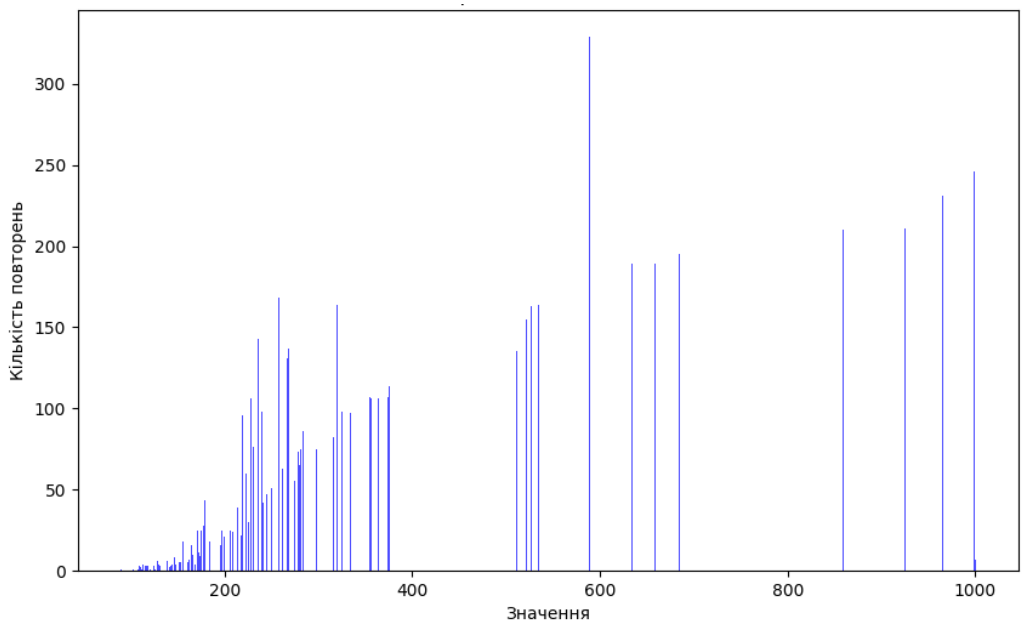

Рис.3.9 Генерована послідовність на основі спектрального аналізу

У даному випадку виконання функції стандартизації є ключовою, особливо якщо для користувача є важливим різноманітність генерованих чисел, а не чистота їх отримання з аудіосигналу (Рис.3.10).

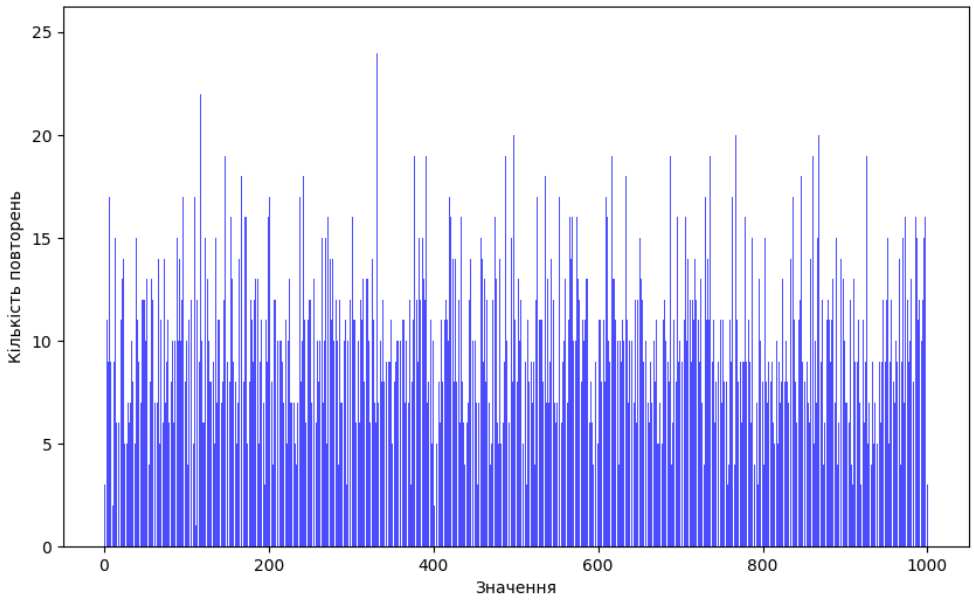

Рис.3.10 Генерована послідовність на основі спектрального аналізу після стандартизації

3) Середня амплітуда з використанням передекспозиції

При виборі аналогічних меж та кількості елементів перевіримо роботу системи на основі середньої амплітуди з використанням передекспозиції (Рис.3.11).

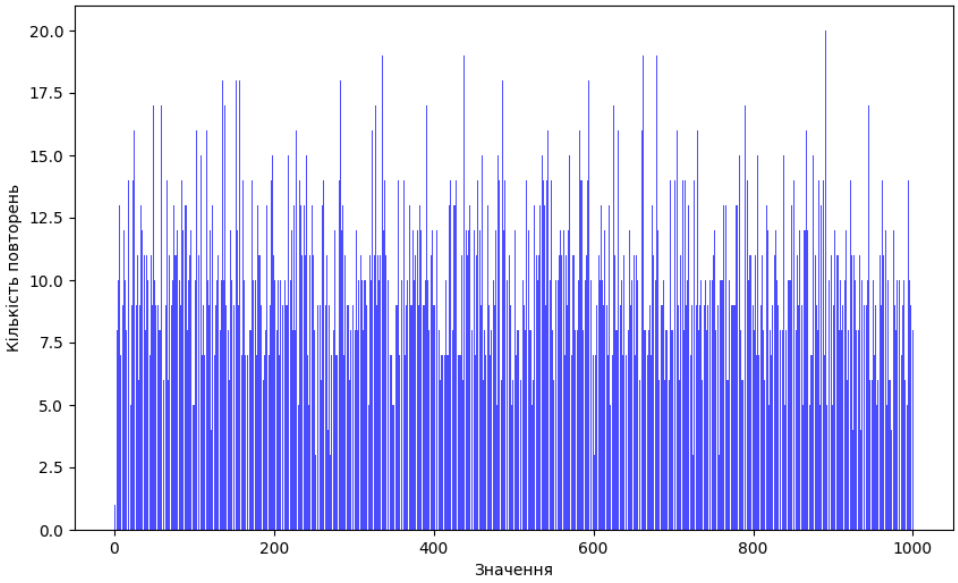

Рис.3.11 Генерована послідовність на основі середньої амплітуди з використанням передекспозиції

Аналогічно до використання першого аудіопараметра (середня амплітуда), генерація послідовності на основі середньої амплітуди з використанням передекспозиції також демонструє високий рівень різноманітності генерованих елементів та не має високих піків, що вибиваються із послідовності. Незважаючи на це, програма працює і з цією послідовністю, демонструючи стандартизовану послідовність (Рис. 3.12), а користувач може вирішити, яка саме послідовність є більш доцільною для нього, залежно від того, для вирішення яких задач та цілей він генерує числа.

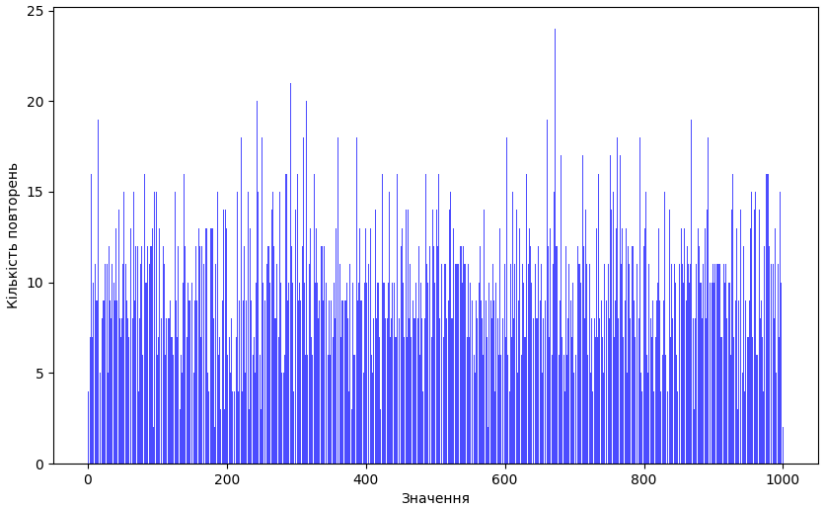

Рис.3.12 Стандартизована послідовність на основі середньої амплітуди з використанням передекспозиції

4) Середня амплітуда з використанням передекспозиції та випадковий зсув

При виборі аналогічних меж та кількості елементів перевіримо роботу системи на основі середньої амплітуди з використанням передекспозиції з додаванням випадкового зсуву (Рис.3.13).

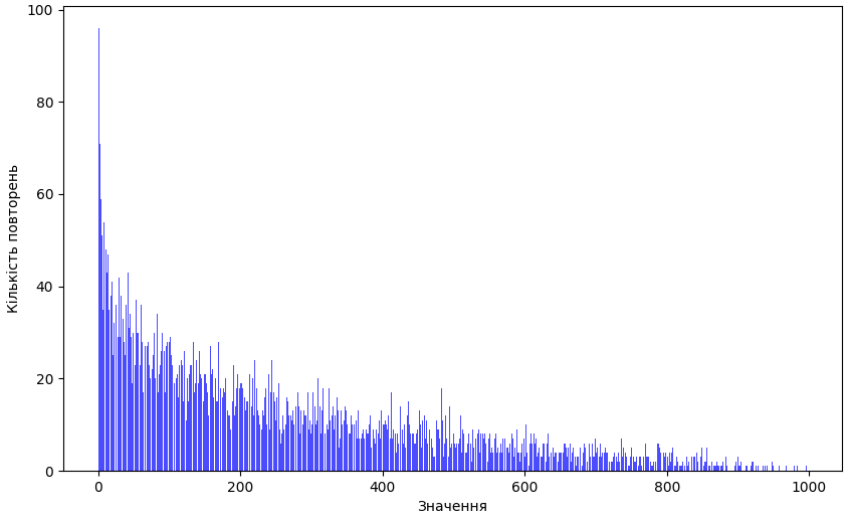

Рис.3.13 Генерована послідовність на основі середньої амплітуди з використанням передекспозиції та випадкового зсуву

Даний аудіопараметр при використанні його в генерації послідовності демонструє спадний розподіл, при якому нижня межа має найбільшу

кількість повторень, а усі наступні елементи мають меншу кількість повторень та ця кількість зменшується при наближенні до верхньої межі.

Якщо для користувача є неприйнятним такий розподіл для задач, що він вирішує, стандартизація генерованої послідовності (Рис.3.14) допоможе йому використати цей аудіопараметр для генерації.

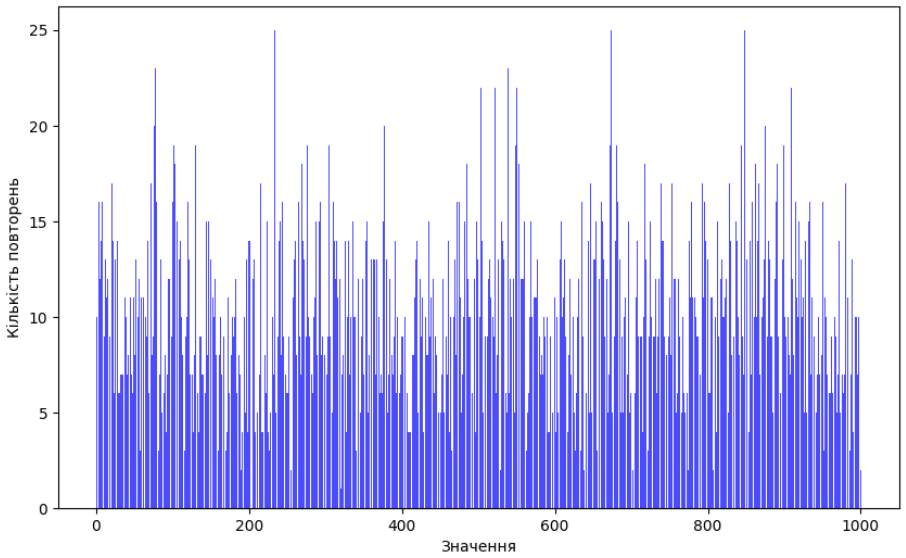

Рис.3.14 Стандартизована послідовність на основі середньої амплітуди з використанням передекспозиції та випадкового зсуву

5) Характеристика аудіосигналу MFCC

При виборі аналогічних меж та кількості елементів перевіримо роботу системи на основі характеристики аудіосигналу MFCC (Рис. 3.15).

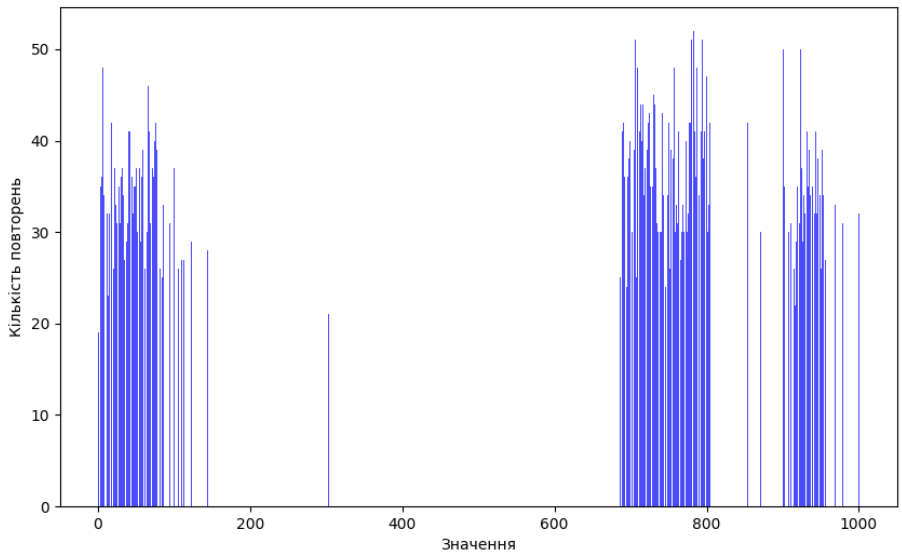

Рис.3.13 Генерована послідовність на основі MFCC

Генерована послідовність демонструє великі прогалини між значеннями діапазону, в яких вони знаходяться, що робить використання даного аудіопараметра MFCC для генерації специфічним та підходить тільки для задач, де генерація істинно випадкових чисел є більш значущою, ніж рівномірність розподілу. Проте функція стандартизації вирішує проблему рівномірності та демонструє результат (Рис. 3.14), що буде прийнятним для задач генерації, що потребують відсутності прогалин.

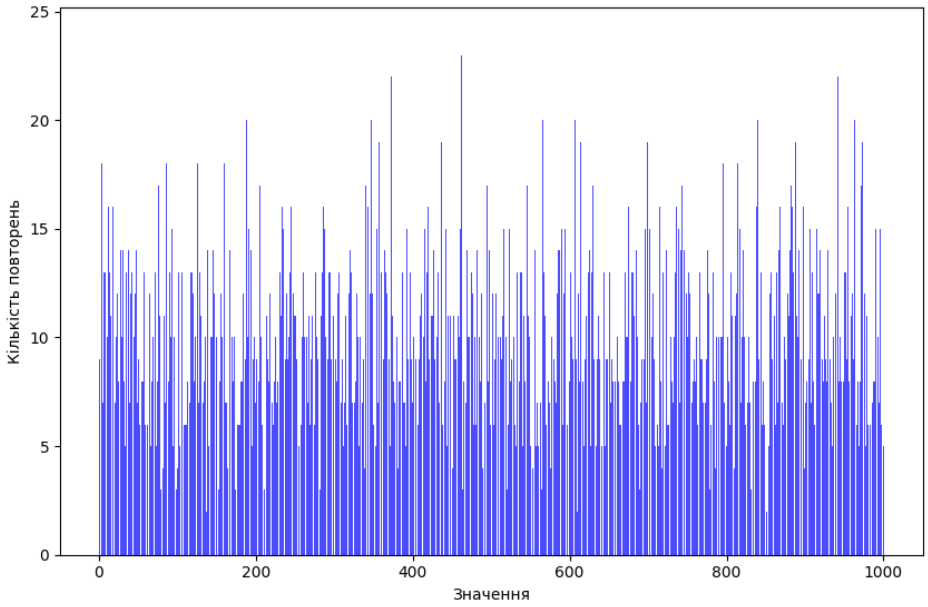

Рис.3.14 Стандартизована послідовність на основі MFCC

При порівнянні генерованих послідовностей на основі різних аудіопараметрів можна визначити, що кожен із них може використовуватися для розв’язку різних задач, залежно від того, яку характеристику генерованої послідовності вважати критичною: істинна генерація випадкового числа, рівномірність розподілу та ін.

Функція стандартизації послідовності демонструє виключно стабільний результат, тому генеровані послідовності, що проходять через неї, можуть використовуватись для розв’язку більшості прикладних задач, що потребують генерації випадкових чисел.

Висновки до розділу 3

У розділі представлено докладний огляд процесу створення та функціоналу розробленої системи генерації випадкових чисел. Застосування аналізу аудіосигналів для отримання числових послідовностей виявляється як цікавий підхід, що може знайти застосування в різних областях.

В ході розробки системи були використані інструменти, такі як бібліотеки «sounddevice», «librosa», «numpy», та «matplotlib», що забезпечило необхідні можливості для роботи з аудіоданими та числовими даними, а також для візуалізації результатів.

Розглянуто та реалізовано алгоритми для генерації чисел на основі різних характеристик аудіосигналу, включаючи середнє амплітудне значення, спектральний аналіз, середню амплітуду з використанням передекспозиції та випадкового зсуву, а також характеристики MFCC. Реалізовані функції підрахунку кількості повторень та відображення результатів у вигляді графіків.

Система пройшла тестування для різних аудіопараметрів, результати якого свідчать про різноманітність та ефективність алгоритмів генерації чисел. Графічне представлення генерованих послідовностей дозволяє зрозуміти розподіл отриманих числових значень.

Розроблена система генерації випадкових чисел на основі аналізу аудіосигналів є перспективною, проте вимагає додаткового тестування для PRN. Результати демонструють ефективність використання аудіосигналів для створення псевдовипадкових послідовностей, але важливо провести більш глибокий аналіз статистичних та криптографічних властивостей згенерованих чисел.Додаткове тестування повинно включати аналіз розподілу, періодичності та незалежності PRN.

ВИСНОВКИ

Магістерська робота присвячена розробці системи генерації випадкових чисел на основі аналізу акустичних сигналів. У процесі виконання досліджень були ретельно розглянуті основні методи генерації псевдовипадкових чисел, включаючи класичні та інноваційні підходи, а також вивчені аспекти оцінки їх якості.

Аналіз акустичних сигналів, проведений у другому розділі, визначив основні характеристики цих сигналів, методи їх обробки та математичні моделі, що відкриває можливості для використання акустичного аналізу у генерації випадкових чисел.

У третьому розділі роботи була успішно реалізована система генерації випадкових чисел на основі аналізу акустичних сигналів. Важливим досягненням є вибір ефективних інструментів розробки, розробка оптимального алгоритму та архітектури системи. Проведене тестування підтверджує функціональність системи та відкриває перспективи для її подальшого використання з виконанням додаткового тестування. Тести на розподіл, автокореляцію та стійкість до криптографічних атак потребують значних досліджень та ресурсів, тому можуть бути проведені у подальших роботах. Додатково, важливо врахувати витрати ресурсів та оптимізацію продуктивності системи.

Отримані результати свідчать про значущість та ефективність використання аналізу акустичних сигналів у сфері генерації випадкових чисел, що робить розроблену систему потужним інструментом для різноманітних додатків, де важлива випадковість та творчий підхід, тому подальші дослідження будуть спрямовані на вдосконалення та оптимізацію системи генерації випадкових чисел.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Blaser, R. (2021). «Random Rotations in Machine Learning.» PhD Thesis, London School of Economics, Department of Statistics, Houghton Street, London WC2A 2AE, United Kingdom. pp [10-22].

2. Blum, Edward K.; Lototsky, Sergey V. (2006). «Mathematics of physics and engineering. World Scientific.» p. 98.

3. Buchanan, William J. (2023). «Linear Congruential Random Number Generator.»URL: https://asecuritysite.com/random/linear.

4. Dolores E. Battle 2012, «Communication Disorders in Multicultural Populations, 4th Edition» DOI: 10.1016/C2009-0-40610-X.

5. Ewert, M. «A Random Number Generator Based on Electronic Noise and the Xorshift Algorithm.» URL: https://d-nb.info/1203628439/34.

6. Gold, B., and Morgan, N. (2000). «Speech and Audio Signal Processing – Processing and Perception of Speech and Music.» John Wiley & Sons, Inc. ISBN: 0471351547.

7. Havelock, David & Kuwano, Sonoko & Vorlaender, Michael. (2009). «Handbook of Signal Processing in Acoustics.» 10.1007/978-0-387-30441-0.

8. Hoang, V.T., Rogaway, P. (2018). «On Generalized Feistel Networks.» Department of Computer Science, University of California, Davis, USA. July 9, 2018.

9. James, F. «A review of pseudorandom number generators.» Computer Physics Communications, vol. 60, 1990, pp. 329–344.

10. Kneusel, R. (2018). «Random Numbers and Computers» (1st ed.). Springer, pp. 13-14.

11. Knuth, Donald (1997). «Seminumerical Algorithms. The Art of Computer Programming. Vol. 2 (3rd ed.)» pp. 10-26.

12. Lenstra, Arjen K. (2011), «Integer Factoring», in van Tilborg, Henk C. A.; Jajodia, Sushil (eds.), Encyclopedia of Cryptography and Security, Boston, MA: Springer US, pp. 611–618, doi:10.1007/978-1-4419-5906-5_455

13. Luizi, P.C.S., Cruz, F.R.B., van de Graaf, J. (June 2010). «Assessing the Quality of Pseudo-Random Number Generator.» Computational Economics, vol. 36, no. 1, pp. 57-67. DOI: 10.1007/s10614-010-9210-6.

14. Ma, X., Yuan, X., Cao, Z. et al.(2016) «Quantum random number generation. npj Quantum Inf 2, 16021.» https://doi.org/10.1038/npjqi.2016.21

15. Marsaglia, G. «A current view of random number generation.» In: Computer Science and Statistics, Proceedings of the 16th Symposium on the Interface. Elsevier Science/North-Holland, 1985, pp. 3-10.

16. NDE Education, Iowa State University. «Acoustic Emission Equipment.» [Електронний ресурс]. URL: https://www.nde-ed.org/NDETechniques/AcousticEmission/AE_Equipment.xhtml

17. P. K. Atrey. (2006) «Audio based event detection for multimedia surveillance.» DOI: 10.1109/ICASSP.2006.1661400

18. P. Nguyen (2010). «Automatic classification of speaker characteristics». International Conference on Communications and Electronics 2010. pp. 147-152. doi:10.1109/ICCE.2010.5670700

19. Pavan, G., Budney, G., Klinck, H., Glotin, H., Clink, D.J., Thomas, J.A. (2022). «History of Sound Recording and Analysis Equipment». https://doi.org/10.1007/978-3-030-97540-1_1

20. Pierce, Allan. (1989). «Acoustics: An Introduction to Its Physical Principles and Applications.» DOI:10.1063/1.2914388.

21. Prasetyowati, S.A.D., Arifin, B., Nugroho, A.A., Khosyi'in, M. (2022). «Exploration of Generator Noise Cancelling Using Least Mean Square Algorithm.» Journal of Electrical Technology UMY (JET-UMY), Vol. 6, No. 1, June 2022 pp. [22-32].

22. Schindler, Werner (2009). «Random Number Generators for Cryptographic Applications». Cryptographic Engineering. Boston, MA: Springer US. pp. 5–23. doi:10.1007/978-0-387-71817-0_2.

23. Zakaria K. (2021) «Every Audio Effect Explained», URL:https://emastered.com/blog/audio-effects-explained

ДОДАТКИ

Додаток А

Вихідний код системи генерації випадкових чисел на основі аудіосигналів

```
import sounddevice as sd
import librosa
import matplotlib.pyplot as plt
import numpy as np
import soundfile as sf
import sys
import random

# 1. Запис звуку
def record_audio(duration=5, fs=44100):
    print("Розпочато запис аудіо...")
    audio_data = sd.rec(int(duration * fs), samplerate=fs,
channels=1)
    sd.wait()
    print("Запис завершено.")
    return audio_data.flatten()

# 2. Попередній аналіз
def analyze_audio(audio_data):
    # Перевірка, чи є записаний звук
    if np.all(audio_data == 0):
        print("Звук записано некоректно. Повторіть спробу.")
        input("Натисніть Enter для виходу з програми...")
        sys.exit()  # Вихід з програми
    else:
        audio_trimmed = librosa.effects.trim(audio_data,
frame_length=2048, hop_length=512)[0]
        #Збереження аудіо-даних у файл
        sf.write('analyzed_audio.wav', audio_trimmed, 44100)
        print("Записані аудіо-дані збережено у файл
'analyzed_audio.wav'.")
        return audio_trimmed
```

```
# 3. Введення користувачем діапазону
def get_user_input():
    lower, upper = None, None

    while lower is None or upper is None or lower > upper or
lower == upper:
        try:
            lower = int(input("Введіть нижню межу діапазону: "))
            upper = int(input("Введіть верхню межу діапазону:
"))
            if lower > upper:
                print("Нижня межа діапазону повинна бути меншою
за верхню.")
                lower, upper = None, None
            elif lower == upper:
                print("Межі діапазону не можуть бути
однаковими.")
                lower, upper = None, None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть цілі
числа.")
            lower, upper = None, None

    num_generated = None
    while num_generated is None:
        try:
            num_generated = int(input("Введіть бажану кількість
генерованих чисел: "))
            if num_generated <= 0:
                print("Недопустиме значення. Кількість має бути
більше за 0.")
                num_generated = None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть ціле
додатнє число.")
```

```
            num_generated = None

    audio_param_type = None
    while audio_param_type is None:
        try:
            print("")
            audio_param_type = int(input("""
Параметри для генерації:
    1 - Середнє амплітудне значення аудіосигналу
    2 - Спектральний аналіз
    3 - Середня амплітуда з використанням передекспозиції
    4 - Середня амплітуда з використанням передекспозиції та
випадковий зсув
    5 - MFCC
Оберіть параметр: """))
            if audio_param_type <= 0 or audio_param_type > 5:
                print("Недопустиме значення. Тип параметру має
бути в межах від 1 до 5.")
                audio_param_type = None
        except ValueError:
            print("Неправильний формат. Будь ласка, введіть ціле
додатнє число.")
            audio_param_type = None

    return lower, upper, num_generated, audio_param_type

# 4. Генерація чисел на основі аудіосигналу
def generate_numbers(audio_data, user_input):
    lower, upper, num_generated, audio_param_type = user_input

    generated_sequence = []
    element_counts = {}  # словник для збереження кількості
повторень кожного елементу
    mean_value = np.mean(audio_data) #середнє відхилення
```

```
    # Спектральний аналіз
    spec = librosa.feature.melspectrogram(y=audio_data,
sr=44100, n_fft=2048, hop_length=512)

    # Отримання амплітуд аудіоданих
    amplitude =
np.mean(np.abs(librosa.effects.preemphasis(audio_data)))

    # Отримання MFCC для аудіоданих
    mfcc = librosa.feature.mfcc(y=audio_data, sr=44100,
n_mfcc=13)

    for _ in range(num_generated):
        if audio_param_type == 1:
            # Генеруємо числа на основі середнього амплітудного
значення аудіосигналу
            generated_number = int(np.random.normal(mean_value,
num_generated / 3))

            # Поки згенероване число дорівнює lower-1 або
upper+1, генеруємо нове
            # Для виключення великої кількості елементів, що
дорівнюють межам
            while True:
                generated_number = max(min(generated_number,
upper + 1), lower - 1)
                if generated_number != lower - 1 and
generated_number != upper + 1:
                    break
                generated_number =
int(np.random.normal(mean_value, num_generated / 3))

        elif audio_param_type == 2:
            # Вибираємо випадково стовпці
```

```
            column_index = random.sample(range(spec.shape[1]),
10)

            # Отримуємо значення вибраного стовпчика
            selected_column = spec[:, column_index]

            # Нормалізуємо значення стовпчика до діапазону
[lower, upper]
            generated_number = int(
                (np.max(selected_column) -
np.min(selected_column)) / (np.max(spec) - np.min(spec)) * (
                                upper - lower) + lower)
        elif audio_param_type == 3:
            # Генеруємо число в межах [lower, upper] на основі
середньої амплітуди
            # Отримуємо числа в межах [0:1]
            generated = lower + np.random.rand() * (upper -
lower) * amplitude
            # Визначаємо, скільки розрядів використовувати для
кожного числа
            scale_factor = 10 ** 5

            # Створюємо нові числа в межах вказаного діапазону
            scaled_numbers = lower + (generated * scale_factor)
% (upper - lower)

            # Забезпечуємо, що значення знаходяться в межах
            generated_number = np.round(np.clip(scaled_numbers,
lower, upper)).astype(int)
        elif audio_param_type == 4:
            # Генеруємо випадковий зсув в межах
            shift = np.random.uniform(lower, upper)

            # Масштабуємо значення на основі середньої амплітуди
та випадкового зсуву
```

```
            scaled_number = int(np.round((shift / amplitude) *
np.random.uniform(0, amplitude)))

            # Забезпечуємо, що значення знаходиться в межах
            generated_number = min(upper, max(lower,
scaled_number))
        else:
            # Нормалізуємо MFCC до діапазону [0, 1]
            normalized_mfcc = (mfcc - mfcc.min()) / (mfcc.max()
- mfcc.min())

            # Перетворюємо MFCC у вектор
            mfcc_vector = normalized_mfcc.flatten()

            # Генеруємо числа від 0 до 1 на основі MFCC
            generated_numbers = np.random.choice(mfcc_vector,
num_generated)

            # Підганяємо числа до вказаного діапазону
            generated_numbers_scaled = np.round(lower +
generated_numbers * (upper - lower)).astype(int)

            # Отримуємо унікальні числа
            unique_numbers = np.unique(generated_numbers_scaled)

            # Випадковий вибір чисел зі списку унікальних чисел
            selected_numbers = np.random.choice(unique_numbers,
num_generated)

            # Випадковий вибір одного числа з вибраних чисел
            generated_number =
np.random.choice(selected_numbers)

        generated_sequence.append(generated_number)
        # Підрахунок кількості повторень елементу
```

```
        if generated_number in element_counts:
            element_counts[generated_number] += 1
        else:
            element_counts[generated_number] = 1 #якщо
зустрічається вперше
    with open("generated_sequence.txt", "w") as file:
        file.write(', '.join(map(str, generated_sequence)))

    return generated_sequence, element_counts

def standardize_sequence(sequence, user_input):
    lower, upper, num_generated, _ = user_input

    # Стандартизація до [0, 1]
    min_value = np.min(sequence)
    max_value = np.max(sequence)
    standardized_sequence = (sequence - min_value) / (max_value
- min_value)

    # Функція кумулятивної ймовірності
    cumulative_prob = np.cumsum(standardized_sequence) /
np.sum(standardized_sequence)

    # Згенеруємо випадкові числа для нової послідовності
    random_values = np.random.rand(num_generated)

    # Застосуємо функцію кумулятивної ймовірності для отримання
нових значень
    standardized_sequence = np.round(np.interp(random_values,
cumulative_prob, np.linspace(lower, upper,
len(sequence)))).astype(int)

    # Підрахунок кількості повторень для кожного елементу
    element_counts = {}
    for number in standardized_sequence:
```

```
        if number in element_counts:
            element_counts[number] += 1
        else:
            element_counts[number] = 1
    # Збереження стандартизованої послідовності
    with open("standardized_sequence.txt", "w") as file:
        file.write(', '.join(map(str, standardized_sequence)))

    return standardized_sequence, element_counts

# 6. Гісторама значень генерованої послідовності
def display_element_counts(element_counts,user_input,
is_normalized):

    lower, upper, num_generated, audio_param_type = user_input

    if audio_param_type == 1:
        param_type = "Середнє амплітудне значення аудіосигналу "
    elif audio_param_type == 2:
        param_type = "Спектральний аналіз"
    elif audio_param_type == 3:
        param_type = "Середня амплітуда з використанням
передекспозиції"
    elif audio_param_type == 4:
        param_type = "Середня амплітуда з використанням
передекспозиції та випадковий зсув"
    else:
        param_type = "Характеристика аудіосигналу MFCC"

    elements = list(element_counts.keys())
    counts = list(element_counts.values())

    plt.figure(figsize=(10, 6))
    plt.bar(elements, counts, color='b', alpha=0.7)
    plt.xlabel('Значення')
```

```
    plt.ylabel('Кількість повторень')
    plt.gcf().canvas.manager.window.wm_title(f'Межі:
[{lower},{upper}], Кількість чисел: {num_generated}, Тип
параметра: {param_type}')

    if is_normalized:
        plt.title('Генерована послідовність після
стандартизації')
        plt.show()
    else:
        plt.title('Генерована послідовність')
        plt.show(block=False)  #для виведення обох графіків
одночасно
        plt.pause(1)  #графіки не встигають побудуватись та
підвисають без затримки

# Головна функція
if __name__ == "__main__":
    audio_data = record_audio()  # 1. Запис звуку
    audio_trimmed = analyze_audio(audio_data)  # 2. Попередній
аналіз
    user_input = get_user_input()  # 3. Введення інформації
користувачем
    generated_sequence,  element_counts =
generate_numbers(audio_trimmed, user_input)  # 4. Генерація
чисел
    standardized_sequence_result, standardized_element_counts =
standardize_sequence(generated_sequence, user_input)
#Стандартизація послідовності
    display_element_counts(element_counts, user_input,
is_normalized=False)  # Побудова графіку без нормалізації
    display_element_counts(standardized_element_counts,
user_input, is_normalized=True)  # Побудова графіку з
нормалізацією
```