

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

"_____ " _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ Лішук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА
МАГІСТРА

Програмна модель системи локалізації звукових джерел

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки

Пиндик Дмитро Анатолійович

Науковий керівник – д.т.н., професор

Кулаков Павло Ігорович

Умань – 2023

Реферат
на кваліфікаційну роботу «Програмна модель системи локалізації
звукових джерел»

студента факультету економіки та підприємництва
Уманського національного університету садівництва

Пиндика Дмитра Анатолійовича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 70 сторінок, 28 рисунків, 4 додатка, використано 45 літературних джерел.

Кваліфікаційна робота магістра присвячена розробці програмної моделі системи локалізації звукових джерел. Проведено аналіз предметної області, в результаті якого було систематизовано недоліки аналогічних програмних моделей та систем, розроблена архітектура програмних засобів для моделювання системи аудіолокації, розроблено програмне забезпечення для програмної моделі системи локалізації звукових джерел, проведено його тестування. Об'єктом дослідження є програмне забезпечення для процесу моделювання системи локалізації звукових джерел. Метою дослідження є підвищення точності аудіолокації шляхом розроблення та застосування засобів моделювання системи, що реалізує алгоритм із використанням кореляційної обробки сигналів. Наукова новизна отриманих результатів полягає в удосконаленні алгоритмів аудіолокаційного пошуку, що дозволяє на основі застосування кореляційної функції та коригування підвищити точність цього процесу. Практичне значення одержаних результатів отриманих результатів полягає в можливості використання результатів роботи для військових потреб.

У першому розділі кваліфікаційної роботи магістра проведено аналітичний огляд предметної області, у другому розділі розроблено архітектуру програмних засобів для моделювання системи аудіолокації, у третьому розділі розроблено програмне забезпечення моделі системи локалізації звукових джерел.

Анотація

Кваліфікаційна робота магістра присвячена розробці програмної моделі системи локалізації звукових джерел, розглянуто та модифіковано метод покращення розпізнавання напрямку до джерела звуку. За допомогою середовища розробки Visual Studio, а також мови програмування C#, проведено розробку програмного застосунку, який використовує оцифровані дані звукового сигналу для моделювання процесу локалізації джерела звуку. Окрім того програмне забезпечення дозволяє виконувати моделювання процесу визначення напрямку до джерела із зміною вхідних даних, таких як частота дискретизації сигналу, тип конфігурації приймача, масштаб, кількість та позиція джерел звуку, або приймачів.

Ключові слова: програмна модель, локалізація, джерела звуку, інформаційне забезпечення.

Annotation

The master's qualification work is devoted to the development of a software model of the sound source localization system, the method of improving the recognition of the direction to the sound source is considered and modified. Using the Visual Studio development environment, as well as the C# programming language, a software application was developed that uses digitized sound signal data to simulate the sound source localization process. In addition, the software allows you to simulate the process of determining the direction to the source by changing the input data, such as the sampling rate of the signal, the type of receiver configuration, the scale, number and position of sound sources or receivers.

Keywords: software model, localization, sound sources, information support.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра Інформаційних технологій

Освітня програма Комп'ютерні науки

Спеціальність 122 Комп'ютерні науки

(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

Ліщук Р.І.

(підпис)

(ініціали, прізвище)

" _____ " _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту Пиндику Дмитру Анатолійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту Програмна модель системи локалізації звукових джерел

керівник роботи Кулаков Павло Ігорович, д.т.н., професор кафедри

інформаційних технологій

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "28" квітня 2023 року № 01-06/77

2. Строк подання студентом роботи 20.12.2023

3. Вихідні дані до роботи Розробити програмну модель системи локалізації звукових джерел, провести розробку програмного застосунку моделювання процесу локалізації джерела звуку та моделювання процесу визначення напряму до джерела із зміною вхідних даних

4. Перелік питань, які потрібно розробити аналітичний огляд предметної області, архітектура програмних засобів для моделювання системи аудіолокації, програмне забезпечення моделі системи локалізації звукових джерел.

5. Перелік графічного матеріалу 28 рисунків, 4 додатка, 45 джерел

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи	05.09.2022	
2.	Одержання індивідуального завдання	07.09.2022	
3.	Складання календарного плану написання кваліфікаційної роботи	08.09.2022	
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи	Впродовж I семестру	
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи	До початку II семестру	
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків	09.02.2023 23.03.2023 26.04.2023 10.05.2023	
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника	18.05.2023	
8.	Написання та оформлення роботи в остаточному варіанті	25.05.2023	
9.	Перевірка роботи на академічний плагіат	12.06.2023	
10.	Одержання відгуку наукового керівника	13.06.2023	
11.	Попередній захист роботи на кафедрі	14.06.2023	
12.	Затвердження роботи завідувачем кафедри	15.06.2023	
13.	Одержання рецензії	16.06.2023	
14.	Захист кваліфікаційної роботи	19-21.06.2023	

Студент _____ **Пиндик Д.А.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Кулаков П.І.**
(підпис) (прізвище та ініціали)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....	11
1.1. Аналіз систем аудіолокації та методів їх реалізації	11
1.2. Аналіз недоліків та переваг існуючих систем для визначення місцезнаходження джерела звуку.....	20
1.3. Постановка задачі.....	28
РОЗДІЛ 2. АРХІТЕКТУРА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ МОДЕЛЮВАННЯ СИСТЕМИ АУДІОЛОКАЦІЇ.....	29
2.1. Метод локалізації на пряму до джерела звуку	29
2.2. Корегування результатів моделювання	32
2.3. Розробка оптимальної архітектури кластерів.....	35
РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛІ СИСТЕМИ ЛОКАЛІЗАЦІЇ ЗВУКОВИХ ДЖЕРЕЛ.....	39
3.1. Розробка програмного забезпечення моделі системи локалізації звуків джерел.....	39
3.2. Створення та реалізація інтерфейсу користувача	48
3.3. Тестування програмної моделі системи.....	53
ВИСНОВОК.....	57
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	59
ДОДАТОК А Результати розрахунків ВКФ для зміщених сигналів.....	63
ДОДАТОК Б Фрагмент лістингу програмного забезпечення.....	64
ДОДАТОК В Розроблені конфігурації.	69
ДОДАТОК Г Вигляд інтерфейсу користувача.....	70

Перелік умовних позначень, скорочень і термінів

ВКФ	Взаємкореляційна функція
БПЛА	Безпілотний літальний апарат
ІТ	Інформаційні технології
ІС	Інформаційна система
ПЗ	Програмне забезпечення
ПД	Програмний додаток

Вступ

Звук як фізичне явище та його використання є невичерпною темою для досліджень, оскільки він є базовою складовою сенсорної системи людини. Так, тільки в 1850-х роках звук було вперше записано, а вже в 1870-х звук навчилися передавати, у 1880-х вдалося виконати фотофіксацію звукових хвиль. Із того часу проводилися різноманітні дослідження, які дозволили вивчити фізичні властивості звуку, їх вплив на неживі об'єкти та організми, в результаті чого утворилася окрема наука — акустика, що вивчає поширення звуку в різних середовищах. Лише через 100 років людство почало використовувати звук як інструмент. Так, під час ультразвукового дослідження використовується звук зі спеціальною частотою. У разі зведення нових будівель звук може використовуватися для визначення слабких місць об'єктів, що будуються. В біології, екології, психології, філософії проводяться різноманітні дослідження за допомогою саме звуку. Сьогодні звук використовується у військовій справі із застосуванням методів звукометрії. Звукометрія (або звукова розвідка) — спеціальний вид розвідки, метою якого є знаходження об'єктів по звуку, який вони утворюють.

Актуальність теми дослідження обумовлена тим, що незважаючи на потребу різних країн світу в даного виду системах, лише кілька корпорацій створюють звукометричні системи, які, до того ж, є дорогими, складними в обслуговуванні та користуванні, що унеможлиблює їх широке застосування для оптимізації військових операцій в світі взагалі, та в Україні, зокрема. Разом із тим, створення та подальше впровадження конкурентноспроможної звукометричної системи допоможе нашим військовим визначити локацію загроз та їх ліквідувати. Це зумовлює потребу розроблення власної аудіолокаційної системи, що має бути позбавлена окреслених вище недоліків.

Об'єкт дослідження кваліфікаційної роботи магістра - програмне забезпечення для процесу моделювання системи локалізації звукових джерел.

Предмет дослідження кваліфікаційної роботи магістра — методи та

засоби моделювання систем локалізації звукових джерел.

Метою дослідження кваліфікаційної роботи магістра є підвищення точності аудіолокації загроз шляхом розроблення та застосування засобів моделювання відповідної системи, що реалізує модифікований алгоритм із використанням взаємкореляційного оброблення оцифрованих сигналів.

Задачі дослідження:

- проаналізувати недоліки та переваги існуючих звукометричних методів;
- на базі проведеного аналізу запропоновано покращений алгоритм аудіолокації із використанням взаємно кореляційної функції;
- провести математичне моделювання такого алгоритму для підвищення точності отриманих даних;
- розробити ПЗ, що уможливило комп'ютерну реалізацію системи аудіолокації на базі принципів об'єктно-орієнтованого програмування, що використовує оцифровані дані сигналів для розрахунків і моделювання результатів роботи алгоритму;
- виконати тестування запропонованої системи та її порівняння з аналогами, що представлені на ринку.

Методи дослідження:

- методи статистичного аналізу отриманих результатів – із метою зменшення похибки знайдених результатів;
- методи оцифровування сигналів – для локалізації загроз на основі створення відповідної системи аудіолокації;
- метод об'єктно-орієнтованого програмування для розроблення надійної комп'ютеризованої системи аудіолокації.

Наукова новизна отриманих результатів полягає в удосконаленні алгоритмів оброблення даних аудіолокаційного пошуку, що на відміну від існуючих підходів, дозволяє на основі застосування взаємно кореляційної функції та подальшого її математичного коригування підвищити точність такого процесу.

Практичне значення одержаних результатів полягає в можливості використання результатів моделювання систем локалізації джерел звуку для військових потреб.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз систем аудіолокації та методів їх реалізації

Одним із методів попередження загрози життю та здоров'ю людини є аудіолокація. Розпізнавання звуку може полегшити роботу службам правопорядку або військовим на передовій. Спеціалізовані звукометричні системи також можна використовувати разом з охоронними системами з метою локалізації місця події і мінімізувати наслідки надзвичайної ситуації [1-5]. Отже, у цьому розділі проаналізовано етапи розвитку звукометрії, роботи найвагоміших дослідників у цій галузі, проаналізовано існуючі звукометричні системи, виявлено їх переваги та недоліки, вивчено метод кінцевих різниць та його використання, виконано адаптацію алгоритму локалізації до поставлених у магістерській роботі задач, запропоновано мінімізувати кількість розрахунків за допомогою використання взаємно кореляційної функції.

Основи методів аудіолокації було закладено у 1844 році на Гаврському полігоні, що знаходиться у Франції. Там проводили досліди, спостерігаючи за польотом снарядів, звуком, який вони утворювали. В результаті спостережень було виявлено балістичні хвилі, інша назва яких — хвилі Маха [6-8].

Через майже 50 років, а саме у 1891 році на Гаврському полігоні розроблялися та тестувалися методи, основною задачею яких був пошук швидкості польоту снаряду, завдяки використанню звуку. Даний випадок вважається першим задокументованим випадком використання звуку армією.

У 1904 — 1905 рр. відбувся стрибкопо-дібний розвиток аудіолокації. Причиною цьому послужила конструктивна недосконалість тодішньої артилерії, яка не могла стріляти на великі відстані, а тому змушена була розташовуватися неподалік від передової, тим самим виказуючи свою позицію ворогу. Під час наведення на таку ціль важливо було знати лише відстань до цілі. Зазвичай вирахувати її було не складно, використовуючи метод спостереження та секундомір. Під час пострілу засікали час від спалаху вистрілу, до часу надходження звуку до спостерігача. Далі, використовуючи час надходження та

швидкість звуку, пошук відстані ставав простим. Проте знайти артилерію, яка була замаскована в рослинності чи за будівлями, або ж стояла подалі від передової стало неможливо, оскільки не можна було визначити напрям до артилерії і обчислити дальність на його основі, адже не було спалахів, на які можна було б орієнтуватися, все що залишалось із вхідних даних — лише звук від пострілу [9, 10].

В основі звукометрії лежить метод кінцевих різниць, для використання якого застосовують різні підходи [11-15]. Найпростішим методом інтерполяції є метод скінченних різниць у часовій області, він є одним із найбільш популярних через свою простоту, а також можливість виконати обрахунки для великої кількості різноманітних задач. Основа його полягає у дискретизації функції на певній обчислювальній сітці і її математичний розрахунок шляхом розв'язання рівнянь Максвела за допомогою використання алгоритму, розробленого науковцем Кейном Ші-Гон Йі.

Метод скінченних різниць у часовій області, або FDTD method (Finite-difference time — domain method) — техніка цифрового аналізу, що полягає у визначенні результату диференціальних рівнянь, а також використовується для електромагнітного моделювання або моделювання взаємодії електричних хвиль з оточенням та фізичними об'єктами [16-18].

Даний метод є ефективним, універсальним та бездивергентним. У ньому також може бути виконана проста декомпозиція по даним, побудова радіолокаційних портретів об'єктів, але такі переваги потребують досить значних обчислювальних ресурсів, хоч і не таких значних, як інші алгоритми, що вимагають ще більшої кількості обчислень.

Для розрахунку методу кінцевих різниць необхідно виконати такі дії:

- задати область розрахунків, сітку, граничні умови, що відповідають умові задачі і можуть бути періодичними або поглинаючими;
- задати джерело хвиль;
- згенерувати електромагнітну хвилю, яка є кінцевою у часі;

- перевести в частотне представлення значень полів із використанням перетворень Фур'є.

На основі методу скінченних різниць було створено програмне забезпечення, що розроблено у роботі. Для цього було проведено його модифікацію таким чином:

- приймати час як лічильну область, відкинути використання сітки і граничних умов, адже програма хоч і візуалізуватиме результат, проте не буде виконувати такі моделювання, які проводяться у інших програмах при використанні стандартного метод скінченних різниць;

- джерелом вхідних даних для програмного відтворення звукових хвиль є набір даних, що програма отримує з файлу та обробляє у подальшому; за необхідності ПЗ можна удосконалити і додати можливість роботи із реальними пристроями введення звукових сигналів;

- файл із вхідними даними з розширенням Wav має в собі запис, який відповідає реальному дискретизованому сигналу, в майбутньому файл можна буде прибрати і використовувати оцифровані дані сигналів, які надходитимуть із пристроїв вводу звукових сигналів;

- використання перетворень Фур'є не є необхідним, оскільки робота виконується разом з оцифрованим звуком, а, отже, маючи вхідні дані та відповідні знання, пов'язані із представленням звуку, достатньо буде використати взаємно кореляційну функцію двох сигналів.

Кореляційна функція встановлює кореляцію у системах на певних просторових координатах. Кореляція є співвідношенням величин, що уможливорює її застосування з метою визначення співвідношення сигналів. Для визначення подібності двох сигналів і співвідношення між ними доцільно використати взаємно кореляційну функцію (ВКФ), що дозволяє визначити, чи один і той самий сигнал обробляється у програмі, чи надходить він із затримкою. Оскільки звук і його програмно модельована копія є фізичним явищем, що не поширюється ментально, то потрібно розуміти, що при надходженні звуку від

джерела до приймача буде затримка, яка залежить від відстані. Оскільки в природі швидкість поширення звуку може коливатися залежно від метеорологічних умов, то для подальшого розгляду роботи прикладу, використаємо значення швидкості 343 м/с або 1236 км/год, що відповідає швидкості поширення звуку у сухому повітрі за температури навколишнього середовища 20°C. Розглянемо ситуацію, в якій джерело звуку знаходиться на відстані 1000 м від приймача. Звук подолає дану відстань за 2,91с. Якщо додати ще один приймач, що віддалений від першого, і дистанція від доданого пристрою до джерела становитиме 1100м, то затримка становитиме вже 3,2с. Маючи затримку та швидкість надходження звуку, можемо визначити можливу позицію, а потім і кут нахилу до його джерела.

Розрахунки, які необхідно було виконати далі, проводилися графічно. Суть їх полягала в тому, щоб розрахувати з отриманих даних 2 гіперболи та знайти їх перетин або ж знайти приблизну позицію із центру акустичної бази. Зазвичай, розв'язок таких задач займав багато часу, але його вистачало для виконання поставленого завдання. Далі, оскільки розрахунки проводилися для артилерії, виконувалися поправки на кут до акустичної бази (пари мікрофонів), швидкість та напрям вітру і відкривався вогонь по приблизно розрахованому розташуванню джерела звуку.

У даній МКР потрібно визначати затримку, обробляючи масиви даних, що надходять із файлу чи з мікрофона. Важливою тут буде дискретизація сигналу. Чим більше її частота, тим точніше результат, і тим більше розрахунків потрібно виконати. Так, наприклад, частота дискретизації 8000 Hz дасть час для одного періоду сигналу 0,000125 с, при використанні частоти 44100 Hz отримаємо проміжок часу 0,00002267573 с, який дасть точніший і ресурсоемніший у 5,512 раз результат. При використанні частоти 192000 Hz, порівняно із 44100 Hz отримаємо приріст у точності та вимогах до системи у 4,4 рази. Тому при використанні більшої частоти необхідно використовувати таке обладнання, яке дозволить швидко обробляти великі масиви даних.

Визначити затримку одного сигналу відносно іншого можна за допомогою ВКФ, формула якої виглядає так:

$$V_{xy}(\tau) = \int x(t) \cdot y(t) \cdot dt, \quad (1)$$

де $x(t)$, $y(t)$ — конкретні дискретні значення сигналів, що надійшли в конкретний проміжок часу t ;

За отриманих вхідних масивах необхідно виконати пошук зсуву сигналу. Розв'язати цю задачу можна, виконуючи зсув між масивами даних та розраховуючи максимум ВКФ для кожного такого зсуву. Зсув виконується таку кількість разів, яку необхідно, на основі розрахунків дистанції між приймачами, швидкості звуку та частоти дискретизації. В результаті отримаємо масив максимумів, найбільший із яких і відповідатиме найбільш можливому зсуву. Після цього можна розрахувати кут і отримати місцезнаходження джерела.

Першим, хто започаткував розвиток звукометрії був Ернест Мах, саме він почав дослідження зі звуком. Все почалося із експериментального дослідження поведінки звукових хвиль. Він проводив фотофіксацію танкових снарядів після пострілу на звукових, дозвукових, надзвукових швидкостях. У результаті експерименту він виявив, що при пострілах дозвукових та звукових, біля снаряду немає чітких деформацій у повітрі, проте, коли снаряд набував надзвукової швидкості, тоді утворювались звукові хвилі, які нагадували конус. Ці хвилі було згодом названо на честь відкривача [19, 20]. Варто зазначити, що це дослідження хоч і не впливає прямо на розвиток аудіолокації, проте воно є початком досліджень, що стосуються звуку, а звукометрія це наука, яка базується на фізичних властивостях звукових хвиль, та їх дослідженнях.

Наступною визначною постаттю у розвитку даної науки є Бенуа К. А. Саме він і вважається засновником звукометрії. Так, він помітив, що під час пострілу людині достатньо просто може визначити напрям до джерела, оскільки є невелика затримка, поки звук надійде до обох вух. Визначити ж позицію з одним для людини неможливо. Якщо ж приймачів звуку буде кілька, як і у людини, і дистанцію між ними збільшити, то можна буде визначати напрям більш точно.

Взявши цей принцип за основу, він почав розробку та тестування системи. В результаті чого, в 1914 році було протестовано прототип в справжніх бойових умовах для наведення артилерії, про що свідчать звіти військових. Також в той же час було задокументовано і приклад донаведення артилерії завдяки звуку вибухів снарядів союзної артилерії. Через те, що розробка вважалася секретною, то автор не надавав ніякої інформації про дослідження та експерименти, які він проводив. Проте, 1935 року він опублікував всі результати роботи, математичні розрахунки розробленого методу знаходження кута до джерела звуку, принцип роботи та результативність винаходу. Незважаючи на такий внесок в розвиток звукометрії, через секретність дослідів про Бенуа, як основоположника звукометрії не було відомо до початку 90-х років.

До таких діячів можна віднести і Густава Фехнера. Він досліджував та описав процес визначення місцезнаходження джерела, як психофізичного процесу. Результатом його роботи стала книга 1860 року, яка складається з двох томів «Елементи психофізики». В ній дослідник представив процес локалізації звуку мозком людини, як такий, що може бути вимірюваним, має взаємозв'язки, які підкоряються чітким фізичним та біологічним законам [21, 22].

Джон Вільям Стретт (лорд Релей) — видатний фізик і основоположник теорії коливань, зробив вклад в експериментальну та теоретичну фізику. До покращень процесу аудіолокації він також має відношення, адже провів експеримент на визначення джерела звуку, який відносився до першої всесвітньо відомої лекції в 1866 р. Він стояв у центрі кола, на колі стояли учасники експерименту та говорили до Джона, він визначав напрямок того, хто говорив. Все це описувалося дослідником математично як фізичне явище завдяки використанню бінаурального співвідношення через відбиття та заломлення звукових хвиль довкола голови людини. Проте, хоч і було просто визначити голос людей, а локалізувати звук камертона у колі було досить складно, особливо низькочастотного [23]. Дослідник вказав, що проблема полягає в частоті звуку. Всі результати проведеного експерименту та подальших досліджень звуку

викладено в книзі «Теорія звуку» (1877 р.), якою досі користуються як посібником інженери та акустики [24 – 25]. Друга її частина має окрему главу, в якій описано факти і теорії слуху. Його друга лекція 1907 р. розкриває і пояснює недоліки розрахунку в першій лекції, що стосувались знаходження камертонів по їх звуку, а також верхній та нижній поріг частот, які здатен розпізнати людський слуховий апарат.

У 1936 р. публікується робота С. С. Стівенса та Едвіна Б. Ньюмена про локалізацію актуальних джерел звуку [26]. В ній автори відштовхуються від локалізації як функції частоти сигналу, функції позиціонування та функції складності, а також подають результати проведених досліджень у графічному та формульному представленнях. Тут наведено інформацію про складність визначення джерела звуку при його знаходженні спереду та ззаду, інформація про що з'являлась і в першій лекції лорда Релея.

Ллойд Джеффрес у 1948 році опублікував свою статтю, де представляє теорію локалізації звуку, засновану на різниці в часі стимуляції двох вух. Гіпотеза залежить від відомої повільної швидкості провідності в дрібних нервових волокнах і явища просторової сумації [27]. Головним, що потрібно зазначити, стала математична модель затримки надходження звуку до вух, а отже, фізіологічні активності моделюються та описуються фізично, що можна використовувати у розрахунках процесу локалізації.

Необхідно відзначити роботу Лю Ц. про локалізацію багатьох джерел звуку з використанням двох мікрофонів [28], адже основа роботи — бінауральна схема оброблення сигналів, що використовується в біологічних системах для визначення позиції джерела звуку за допомогою знаходження інтерауральної різниці часу (interaural time difference, або ITD). Алгоритм оброблення полягає в переведенні вхідних сигналів з двох мікрофонів на частотну область та подальший її аналіз вздовж пар ліній затримки каналів. За допомогою часового інтегрування і нелінійної операції підсилює отримані дані. Завдяки інтегруванню місць збігів визначається азимут джерел звуку. Наводиться аналіз та інформація

про появу пропущених та хибних джерел звуків, проводиться комп'ютерне моделювання, а також наводяться результати роботи програми Вілера.

Варто звернути і на роботи сучасних дослідників, які і надалі розвивають дану науку, наприклад, Бугайова М. із алгоритмом виявлення акустичних сигналів безпілотних літальних апаратів (БПЛА) [29]. Дослідження ґрунтується на основі аналізу звукових хвиль, що надходять від джерела звуку. Сигнал розглядається як сума гармонік, що мають кратні частоти. Задля обробки вхідних сигналів і визначення їх відношення до БПЛА, було використано періодограму. В роботі представлено як послідовна розробка алгоритму, так і результати розробки, результати проведених лабораторних досліджень, аналіз досліджень та методи покращення результатів.

Дослідження Кочан Р. В., Трембач Б. Р. та Трембач Р. Б. про використання стільникового зв'язку для локалізації джерела звуку репрезентують використання фундаментальних наробок звукометрії для використання їх у нових умовах [30]. Так за основу взято принцип артилерійської розвідки, проте використано його задля локалізації джерела звуку шляхом збору даних із автономних сенсорів на місцевості. Передавачі надсилають спеціальні дані, необхідні для розрахунку, із використанням стільникового зв'язку. Коли дані приймаються на пристрій, що зберігає інформацію із набору сенсорів, відбувається розрахунок і на екрані відображається результат. Розрахунки та результати проведеної розробки наведені у роботі.

Ці ж автори провели дослідження похибки аудіолокації джерела звуку, розглядаючи роботу спеціалізованих звукометричних комплексів [31]. Завдяки цьому вони визначили, що апроксимація, яка використовується для спрощення розрахунків, також є джерелом систематичних похибок. Як результат, автори наводять методи поліпшення точності результатів завдяки проведенню додаткових коригувань при певних умовах, як от велика чи занадто мала відстань до джерела. В цьому ж дослідженні наводиться порівняльна характеристика різних звукометричних систем від різних виробників.

Робота Маошена Цзя, Юйсюаня Ву, Чанчуня Бао, і Цзін Вана є публікація результатів досліджень на тему визначення кількох джерел звуків одночасно, завдяки покадровому видаленню компонентів джерела звуку, що домінує [32]. В ній також вирішена проблема падіння точності визначення джерела звуку, що є слабшим. Так в роботі проведено моделювання і лабораторні дослідження процесу визначення кількох джерел. Підтвердження наявності слабшого джерела і прозорості досліджень виконано завдяки статистичній обробці даних.

Гаманн М., Фернандес-Гранде Е., Гунаван Г., Герстофт П. проводили дослідження локалізації джерела звуку із використанням кількох розподілених мікрофонних решіток [33]. Так у ньому застосовано аналіз і оцінку напрямку надходження звуку до мікрофонних решіток. Паралельне відображення між незалежними решітками і джерелом виводиться завдяки набору точок калібрування. Для підвищення точності локалізації використовується проекція результатів.

Кафедра обробки сигналів університету в Любеку, Німеччина, також варта уваги, оскільки її науковці займались вимірюванням звукових полів мікрофонами, які пересуваються у просторі. Так в процесі цього дослідження було використано теорему Найквіста–Шеннона для розрахунків у часовій та просторовій області, просторова інтерполяція та налаштовані лінійні обчислення для реконструкції звукового поля [34].

Дослідники робототехніки із громадського університету в Греноблі, Франція, провели дослідження знаходження роботом у реальному середовищі джерела звуку, який було спотворено шумом та ревербацією [35]. Основним елементом в даному дослідженні є голова робота, в яку вмонтовано два мікрофони, завдяки яким і проводились дослідження. В процесі дослідження було розроблено метод на основі використання відносної функції переносу, як співвідношення між акустичними функціями перенесення сигналів.

Сін Х., Ян С., Цзі С., Ву Х. опублікували роботу в якій представили алгоритм синтезу локалізації пасивного джерела звуку на основі двоплощинного

елемента [36]. Складність визначення такого джерела обґрунтована нечітким значенням, що відповідає розташуванню об'єкта. Для цього необхідно перетворити масив приймачів, який знаходиться в одній площині в масив, що знаходиться в двох площинах одночасно. Під розроблену модель масиву також було розроблено і метод обробки даних на основі алгоритму синтезу. В результаті дослідники отримали похибку в межах 3%.

Заслуговує уваги і дослідження локалізації джерела звуку за допомогою перетворення звукового поля неконформної поверхні на основі сферичної гармонічної декомпозиції [37]. Дослідники використали спеціальну сферичну мікрофонну решітку, яка дозволяє точно визначати кут до джерела звуку у тривимірному просторі. Поверхні як голографії, так і її реконструкції являються конформними. Натомість науковці, запропонували неконформне рішення задачі за допомогою створення матриці передачі на основі розкладання сферичної гармонічної хвилі. Це в свою чергу дозволяє трансформувати сферичну поверхню в циліндричну, спростити розрахунки, підвищити ефективність сферичної мікрофонної решітки.

Отже, базуючись на результатах досліджень вищеперелічених авторів, а також роботах Брюса С. Вілера було запропоновано метод, що розглядається в наступних розділах магістерської роботи.

1.2 Аналіз недоліків та переваг існуючих систем для визначення місцезнаходження джерела звуку

Системи розпізнавання напрямку звуку знайшли найбільшого свого використання саме у військовому напрямку, тому розглянемо надалі системи із цього сектору ринку. Система «Shoulder-Worn Acoustic Targeting System» або «SWATS» QinetiQ широко використовується американською армією. В загальному армії 12 країн закупили 19500 систем, які пройшли бойові випробування у гарячих точках. В армії США система відома під назвою «Індивідуальна акустична система визначення пострілу». До складу цієї системи

входять GPS, акселерометри, мікрофон, компас, гіроскоп [38]. Усі ці модулі вбудовано в накладку, яка фіксується на плечі і дозволяють забезпечити точність по азимуту від мінус $7,5^\circ$ до $+7,5^\circ$ та діапазон значень 10%. До того ж дана система виявилася достатньо надійною, оскільки гарантійні повернення склали лише 1% від усіх проданих екземплярів. Пристрій дозволяє знаходити джерело пострілу, зберігати координати для відстеження переміщення джерела звуку, та передачі даних іншим учасникам загону. Отримані дані можна відстежувати на спеціальному дисплеї вагою 110 грам, або ж на слуховому апараті. QinetiQ NA оновлюють свій асортимент та модифікують попередні розробки, так, відносно недавно, розглянута система має покращені алгоритми та оптимізовано використовує джерела енергії, що дозволило збільшити дистанцію локалізації цілі до 400м і час автономної роботи до 14 годин. Із нових пристроїв даної компанії можна відзначити Ears FSS (Fixed Site System) та Ears VMS (Vehicle Mounted System), які позиціонуються, як системи для оборони бази та оборони транспорту відповідно. Також QinetiQ веде розробку покращеної версії VMS, що може бути використана для прицілювання в бойових модулях, що використовують дистанційне керування, так як відображає тривимірне зображення місця події. Ведеться робота і над новою системою, що буде мати водонепроникний корпус, екран з вищою роздільною здатністю, більш простим меню, покращеним датчиком, який дозволяє обробляти в 50 разів швидше дані, та приріст у 100 раз об'єму пам'яті, який дозволить застосувати більш розвинені алгоритми. Серед недоліків даної системи є те, що ефективна дальність роботи даної системи становить лише 400 м. Компактність та складність системи негативно впливають на розроблення, налаштування та обслуговування системи та кількість часу, який потрібно використати для цього. Ціна висока у зв'язку із специфікою розроблення даної системи для військового сектору, та аспектами, описаними вище.

Ще одним представником в області звукометричних досліджень є компанія Raytheon. Основний її продукт – система «Boomerang». Загалом було продано

більше 10 000 екземплярів представникам різних країн) [38]. Компанія не надає для широкого загалу всю інформацію щодо даної системи і на можливо лише переглянути таблицю-презентацію, розташовану на офіційному сайті компанії. Вона містить характеристики та приклади використання даної системи. Останньою розробкою програми є система Boomerang Warrior-X, яка є модифікацією для піхоти, а також, за певних умов може використовуватися для захисту певного периметру. Вага пристрою становить 6 кг та виглядає як антена із встановленими на неї та направленими в різні сторони мікрофонами, а також невеликий пристрій, що кріпиться на плечі у солдата, виконує ту ж функцію, що і стаціонарний елемент, який може закріплюватись на техніці. Вся система в цілому складається із датчиків, які можна поєднати у мережу і змусити їх працювати як кластер приймачів, синхронізуючи та покращуючи отриманий результат. Дана особливість дозволяє також отримувати результат із врахуванням азимутального та висотного положення, відстані до джерела звуку на 360°. Отримані дані відображаються на карті планшета чи ноутбука і можуть зберігатись чи надсилатись далі при необхідності для інформування чи подальшого аналізу. Датчик, що обробляє дані наведено на рис.1.1



Рисунок 1.1 - Датчик системи Boomerang Warrior-X

Загальний приклад використання системи зображено на рис. 1.2.



Рисунок 1.2 - Приклад використання системи Boomerang Warrior-X

Компанія повідомляє, що система може виявляти не лише постріли куль, а й ракет. За даними тестувань компанії Raytheon, система Warrior-X здатна виявити 90% всіх пострілів, а також розпізнати ракету, частота помилок даної системи, пов'язана із хибним спрацюванням становить менше 1%, а час реакції і надання результатів до 1,5 секунди. Оскільки живлення відбувається через Ethernet кабель, то система працюватиме до моменту її від'єднання від кабелю. Raytheon також займається модифікацією системи для вертольотів, яка називається Boomerang Air, попередня назва «Helicopter Warning and End of Hazard — Acoustic, Acoustic Warning and Hazard Determination for helicopter». Дана модифікація повинна працювати в умовах сильного шуму, який створює гелікоптер, проте вона необхідна, оскільки гвинтокрили є легкою мішенню за низького польоту як для ракет, так і для гвинтівок. Дана система має на меті інформувати пілотів, аби вони могли вчасно зреагувати на загрозу та за необхідності виконати маневр, уникаючи пошкоджень. Boomerang Air використовує покращені програмні алгоритми, щоб уникнути вібраційних та шумових завад. Датчики даної системи (рис. 1.3) успішно вбудовуються в бортову систему літака, а форма датчиків дозволяє замаскувати використання даної системи.



Рисунок 1.3 - Датчик системи Boomerang Air

Про те, як комплектуються вертольоти даною системою, компанія відмовляється надавати інформацію та відомо, що кількість датчиків на борту залежить від виду та моделі гвинтокрила. Також дана компанія розробляє систему, спеціалізовану для автомобілів AAI Textron PDI. Вона представляє собою датчики, що розміщуються на даху транспорту та інтегрується з бортовим комп'ютером транспорту. Таке розташування датчиків дозволяє покрити 360° навколо транспорту, велику азимутальну та висотну точність та кутову похибку приблизно $\pm 1^\circ$, похибка розпізнавання становить до 25% і стає меншою із збільшенням дистанції понад 350 метрів. Максимальна дальність роботи становить 1,2 км. Приклади використання даної системи можна спостерігати на бронеавтомобілі Humvee M1151, система підключена до модуля Crows II. Незважаючи на всі наведені переваги систем компанії, що розробляються чи вже знаходяться на ринку, багато недоліків спричиняються саме перевагами, зокрема, надзвичайна складність як окремих пристроїв, так і всієї системи, її підтримки в робочому стані, налаштуванні, налагодженні взаємодії. Також система розроблена для використання саме в бойових умовах, а максимальна дальність лише 1,2 км, крім того, достатня точність може бути досягнута лише на відстані від 350 м до джерела звуку.

Компанія Асоет-Metravib створила у 2012 році систему Pearl, яка має на меті створити відносно недорогий прилад, який допоможе визначити висоту та

напрям до джерела звуку. Презентація готового прототипу була продемонстрована і можна представити вигляд системи та її функціонування. Системний інтерфейс являє собою набір чутливих стрілок, що під час обертання системи до джерела звуку, змінюють індикацію, що дозволяє знайти точний напрям до джерела звуку. Ця функція є доступною через вбудований процесор та гіроскоп. При безпосередньому наведенні на джерело звуку, індикація стрілок зникає і з'являється зелене коло, що сигналізує про знайдений напрям [39]. У цій системі датчиками є акустична матриця, що побудована із мікрофонів, що направлені в різні сторони для ідентифікації джерела. Завдяки такій конфігурації система здатна розпізнавати надзвукові кулі калібру 5,45 мм – 20 мм. Готовий виріб у корпусі схожий на магазин у пейнтбольному маркері. Даний засіб використовується безпосередньо солдатом, має кріплення під планку Пікатінні, тобто система може вмонтовуватися на кулеметах, гвинтівках та інших типах зброї, що мають такий тип кріплення для прикріплюваних елементів. Система дозволяє відстежувати джерело звуку навіть при зміні позиції бійця до перезавантаження системи, завдяки гіроскопу. Вага всієї системи становить 400 г. За результатами тестувань Metravib отримала такі дані як: ймовірність системи не виявити постріл менше 5%, азимутна та висотна похибка становить $\pm 10^\circ$ і в межах $\pm 20\%$, отримання результату роботи системи становить менше 2-ох секунд. Час автономної роботи при споживанні енергії, меншої за 2 Вт і джерела живлення 1,5 В (дві AA батарежки), становить понад 12 годин. Наявні порти стандарту RS232 чи RS485 і використовуються для інтеграції електронного прицілу та даної системи. Із 1995 року від компанії Metravib доступна ще одна акустична система, яка ввесь час покращувалася. Така система під назвою «Pilarw» здатна ідентифікувати в режимі реального часу постріли із РПГ, гвинтівок, ракет, мінометів та іншої стрілецької зброї при азимутальній похибці в стані спокою та руху $\pm 2^\circ$ та $\pm 5^\circ$ відповідно, та висотній похибці $\pm 5^\circ$. Проте при активній стрільбі похибка даної системи становить від 10% до 20%. Розроблялася дана система для транспорту і чудово вмонтовується в бортові

системи, такі як GPS, що дозволяє надавати більш точні дані про місце знаходження вогневої позиції противника, або є прицільною системою, що змусить реагувати зброю на борту наводитися на джерело звуків пострілів та наносити вогневе ураження. Під час ведення вогню система здатна виявити калібр та попереджує звуковим сигналом про небезпеку (якщо система розпізнала постріл). Дана система працює в двох режимах: міська зона, сільська зона. Відповідно, в урбаністичних умовах краще розпізнає джерело звуку саме режим «міська зона». Система має такі характеристики. Вага 2 кг, після приведення в робочий стан має висоту 50 см, модуль для розрахунку напрямку до цілі та її дистанції становить 3,6 кг. Останнім споживачем даної системи була Франція, яка замовляла цю акустичну систему для модернізації транспортних засобів. Замовлено було всього 80 екземплярів та переобладнано для інтеграції із системою дистанційного ведення вогню Kongsberg Protector M151 для більш точного наведення на ціль. Існує стаціонарна модифікація даної системи, що дозволяє покривати територію площею 1 км x 1 км, використовуючи до 20 датчиків на один розпізнавальний елемент. Дані системи використовуються такими країнами, як Німеччина та Італія, Афганістан тощо. Незважаючи на переваги всіх представників систем компанії, серед недоліків варто виділити те, що система реагує лише на постріли легкого та важкого озброєння. Певні системи ще на стадії прототипу, а ті, що готові до використання, є дорогими і складними для експлуатації та обслуговування.

Відомим представником в області звукометрії є також голландська компанія — Microflown Avisa, основна діяльність якої спрямована на автомобільну промисловість, проте, за допомогою набутого досвіду, компанія змогла розробити надзвичайно компактний і ефективний пристрій в області звукометрії — Acoustic Vector Sensor або AVS [40]. Він використовується для розпізнавання джерел звуку таких, як постріли зброї, наземна та повітряна техніка, артилерія. Особливістю сенсору є те, що він вимірює не тільки тиск, що спричинює звук, а й визначає векторну акустичну швидкість руху частинок (рис.

1.4). Даний датчик використовує технологію MEMS, що дозволяє під час використання двох, розігрітих до 200°C платинових смужок, пропускати повітря, тим самим змінюючи фізичні властивості повітря, що проходить через датчик.

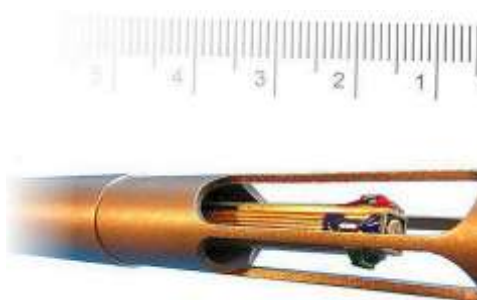


Рисунок 1.4 – Мініатюрний датчик MEMS

Використовуючи дану технологію, можна створити надзвичайно компактний та ефективний сенсор, який важить кілька грам та має розміри (5 мм x 5 мм x 5 мм). Ця технологія дозволяє забезпечити більшу швидкодію і точність порівняно із системами на основі приймачів-мікрофонів, хоча і не здатна визначати висотність джерела. У результаті проведених дослідів було виявлено, що система на основі даних датчиків може ідентифікувати джерело звуку із похибкою 2% або азимутною похибкою $0,5^{\circ}$. Microflow'n Avisa розробила також систему, що використовує дані сенсори і має кодову назву CASTLE. Ця система може встановлюватися як на місцевості, так і на техніку. Результат роботи даної системи наведено на рис. 1.5. До переваг даної системи належать швидкодія, точність та компактність.

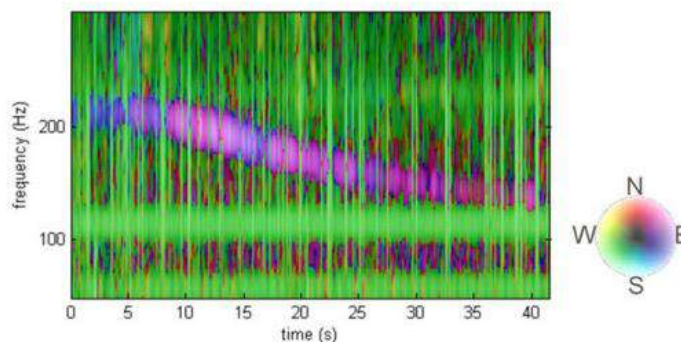


Рисунок 1.5 — Результат розрахунку акустичною системою CASTLE напрямку до джерела

Компанія Ultra Electronics створила схожу на Pearl звукометричну систему під назвою RMGL або Sniper Egg. Дана система здатна розпізнавати джерело надзвукових снарядів, калібром від 5,56мм до 12,7 мм [41]. Вага системи становить 450 грам, має кріплення під планку Пікатінні, що дозволяє її встановлювати на будь-яке озброєння. Дана система має аналогічне до Pearl наповнення та принцип дії і відрізняється тим, що після пострілу на дисплеї розміром 160 пікселів x 128 пікселів із частотою оновлення 10Гц, з'являється стрілка, що вказує напрям. При наведенні солдатом на правильний напрям, з'являється перехрестя, що коригує положення вогнепальної зброї оператора відносно висоти. Із очевидних недоліків системи є, за умови орієнтації на бойові дії, невелика відстань та малий час автономної роботи.

1.3 Постановка задачі

Ціллю магістерської кваліфікаційної роботи є розробка програмної моделі локалізації звукових джерел. Для її реалізації необхідно провести аналіз існуючих систем аудіолокації та методів їх реалізації, здійснити аналіз недоліків та переваг існуючих систем для визначення місцезнаходження джерела звуку.

Розроблена система повинна бути універсальною, мати можливість налаштуватися для локалізації зашумлених звукових джерел на великій дистанції. Система повинна складатися з кластеру приймачів, пристрою для оброблення та програмного забезпечення, програмне забезпечення повинно мати простий і інтуїтивно зрозумілий інтерфейс, який адаптований під симуляцію знаходження звуку і в ньому можна налаштовувати необхідні параметри, такі як частота і відстань.

Завданням розробки є створення програмного забезпечення для моделювання процесу локалізації джерел звуку з використанням системи мікрофонів та засобів аналого-цифрового перетворення сигналів. За потреби повинна забезпечуватись можливість модифікації програмного забезпечення, зміни модельованих мікрофонів, відстані, потужності джерел звуку.

РОЗДІЛ 2. АРХІТЕКТУРА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ МОДЕЛЮВАННЯ СИСТЕМИ АУДІОЛОКАЦІЇ

2.1 Метод локалізації напрямку до джерела звуку

Результат обчислень відображатиметься на консолі, проте спочатку необхідно провести певні розрахунки. На першому етапі розроблення методу локалізації напрямку джерела звуку потрібно визначити зсув сигналів один щодо іншого. Для цього скористаємося взаємнокореляційною функцією — ВКФ (або функція крос-кореляції). Для початку розглянемо масив дискретних даних, показаних на рис. 2.1. Тут представлено фрагменти з 2-х масивів даних (один аудіосигнал, що надходить від різних приймачів). Оскільки сигнал однаковий, його графічне представлення таке саме, але оскільки між мікрофонами є незначна відстань у кластері і звук має подолати її, то для досягнення цілі потрібно певний час. Ця затримка у часі показана на рис. 2.1. При цьому зауважимо, що в реальності кришталево чистих звуків не існує, певні завади будуть завжди. Із цієї причини на графіку два звуки виглядають не однаково, а трохи спотворені один щодо іншого — схожі. Щоб знайти зсув у наборі даних, потрібно вирахувати результат певної функції заданих аргументів (t, t') значень ($x(t), y(t)$), в даному прикладі цими значеннями є масиви.

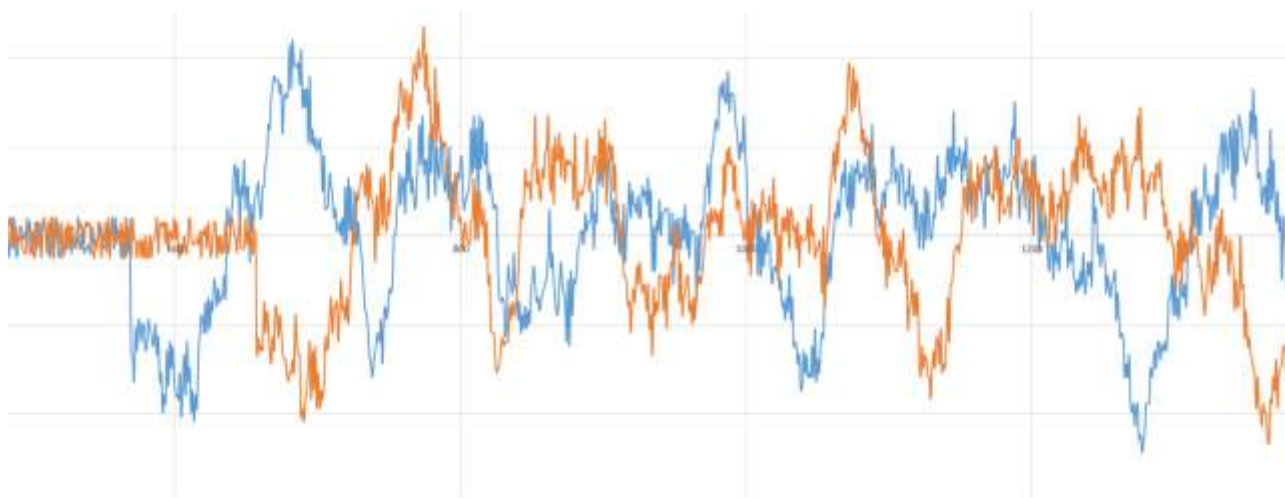


Рисунок 2.1 — Відображення дискретизованих сигналів

$$K_{xy}(t,t')=M[\dot{x}(t)\dot{y}(t')]$$

При розрахунку отримуємо результат, відповідний моменту кореляції для аргументів (t, t') . Якщо результат відмінний від нуля, можна сказати, що масиви корельовані. Отже, ми знайшли один результат, але щоб визначити найточніший момент кореляції, потрібно провести розрахунок $((t-n,t'), \dots, (t-1,t'), (t+1,t'), (t+2,t'), \dots, (t+n,t'))$. Для знаходження достатньої кількості елементів для пошуку, потрібно знати частоту дискретизації, дистанцію між приймачами в кожній мікрофонній парі та швидкість звуку. Це значення розраховується так

$$n=F(l/v),$$

де n - кількість необхідних елементів для розрахунку, l - відстань між мікрофонами, F - частота дискретизації, v - швидкість поширення звуку у просторі.

Однак кількість наборів для перевірки дорівнює $2n$, бо ВКФ обчислює зміщення другого масиву даних відносно першого, а при виконанні обчислень крос-кореляційної функції зі зміщенням першим масивом ми матимемо 0, як результат, тому необхідно виконати перевірку, чи не зсувається перший масив, і якщо так, то на скільки місць в масиві. Оскільки відомо необхідну кількість елементів для перевірки, ми можемо легко розрахувати ВКФ для кожного зсуву масивів. За допомогою програми Microsoft Excel обробляємо результуючі дані розрахунку ВКФ і отримуємо графіки максимумів.

Після розрахунку результатів і введення їх в масив, де індекс певного представленого значення в масиві дорівнює значенню кількості елементів, на яке один сигнал зміщений відносно іншого, а під індексом масиву — результат розрахунку ВКФ, проаналізуємо результати і визначимо відповідне значення. За допомогою методу сортування ми проходимо весь результуючий масив і визначаємо максимальне значення. Індекс максимального елемента результуючого масиву відповідає шуканому значенню.

Якщо визначено зміщення сигналів, швидкість поширення звуку, частоту дискретизації та дистанції у парах мікрофонів, можна визначити шуканий кут до джерела. Розглянемо рис. 2.2, у нас є джерело звуку та 2 мікрофони, які

приймають певний сигнал. На рис. показано, що звук поширюється на більшу відстань до другого мікрофона і тому з'являється затримка. Оскільки

$$\Delta S = l \cdot \cos \alpha,$$

то

$$\cos \alpha = \Delta S / l.$$

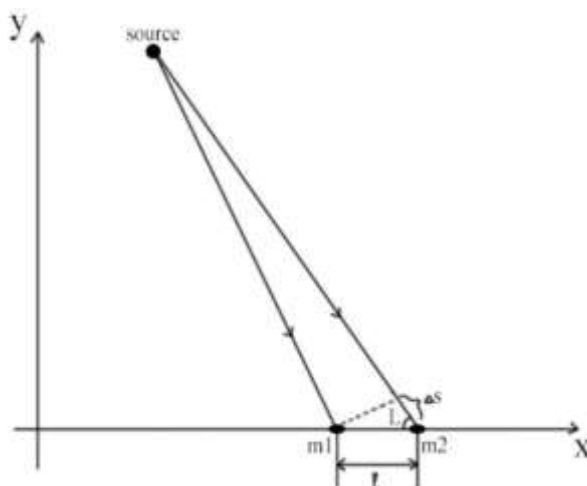


Рисунок 2.2 — Дані, що використовуються для розрахунку кута L до джерела

З урахуванням $\Delta S = v \cdot t$ та $\Delta S = v \cdot T \cdot n$, де $T = 1/F_{\text{discr}}$, формула для розрахунку $\cos \alpha$ буде

$$\cos \alpha = v \cdot n / (l \cdot F_{\text{discr}}),$$

де v - швидкість поширення звуку;

n - індекс найбільшого значення результуючого масиву;

l - дистанція між приймачами у парі;

F_{discr} - дискретизація.

Після обчислення шуканого косинуса кута, скористаємося оберненою тригонометричною функцією арккосинуса, щоб знайти шуканий кут. Оскільки арккосинус є оберненим до функції косинуса, то $\cos y = x$ і $\arccos x = y$. Отже, за допомогою цієї властивості знайдемо шуканий кут до джерела. Зауважимо, що це кут щодо акустичної основи, а не абсолютний. Щоб розрахувати абсолютний, потрібно ввести коригування на кут акустичної основи.

Оскільки в кластерах визначено 4 мікрофони, то кількість акустичних пар становить 6. Розраховані коригування будемо використовувати для подальших розрахунків.

2.2 Корегування результатів моделювання

Варто зауважити, що, насправді, існує 2 можливих кута до джерела звуку. Під час графічного відображення результатів модифікованого методу кінцевих різниць, матимемо 2 кола, центрами яких є приймачі в акустичній парі. Для пошуку джерела звуку знаходимо точку перетину кіл, проте таких точок є дві. Тобто 2 точки на перетині кіл відповідають одному і тому ж куту на рис.2.3.

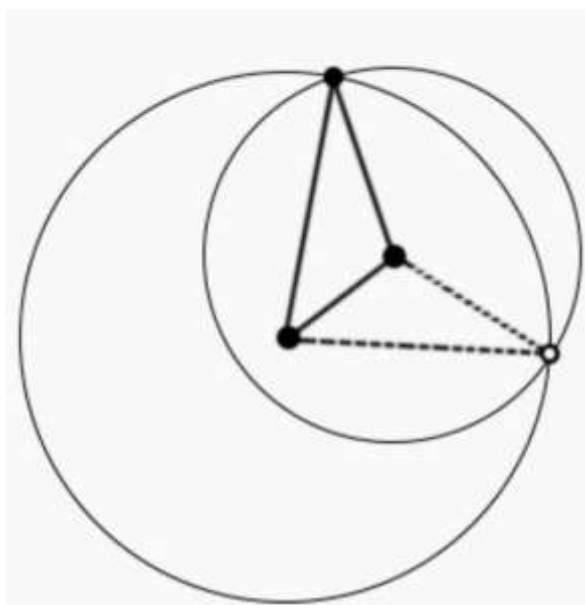


Рисунок 2.3 — Варіант графічного зображення місцезнаходження джерела

Схожа ситуація відображена і на рис.2.4. Зафарбована точка перетину кіл представляє справжнє джерело, звідки вийшов звук, а точка з порожнечою є несправжнім джерелом, що також задовольняє вхідним даним. Один з них є фальшивим, проте одразу невідомо, який саме. Тому потрібно розрахувати кут до другої точки. Для цього достатньо віддзеркалити (інвертувати) відносний кут. Якщо значення дорівнює 60° , а акустична основа має кут 35° до осі ОХ, то інший можливий кут дорівнює 10° , оскільки $60^\circ - 35^\circ = 25$ є відносним кутом, а тому $35^\circ + (-25^\circ) = 10$, де 35° є кут бази, а 25° є зворотним відносним кутом.

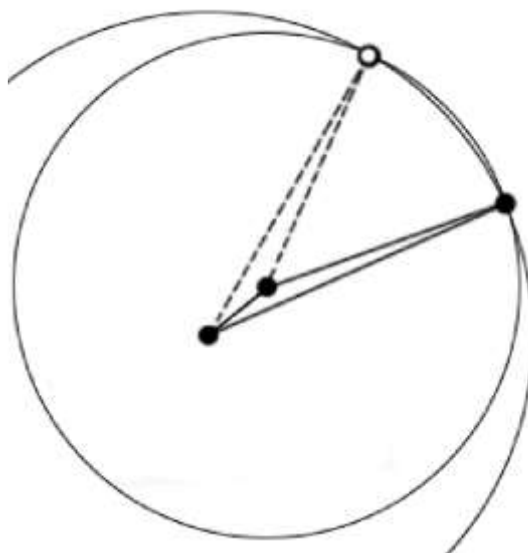


Рисунок 2.4 — Інший варіант графічного зображення
місцезнаходження джерела

Розглянемо підхід до корекції кута. За базу беремо кут відносно акустичної основи. Стосовно бази даний результат є істинним, але щодо глобальної позиції об'єктів — це значення буде швидше за все хибним, наприклад, коли тип кластера не є лінією, оскільки лінія запрограмована бути паралельною до вісі ОХ, тому, наразі, це єдиний тип, який вимагає наульового коригування. Для всіх інших типів коригування розраховується та задається індивідуально.

Слід також зазначити, що помилки можуть накопичуватися у процесі обчислень і це створює досить значну похибку, тому що накопичується значна кількість припущень, прийнятих для полегшення розрахунку. Наприклад, розрахунок арккосинуса, оскільки ця тригонометрична функція не є поступовою. Завдяки цій концепції можна зрозуміти, що різні результати арккосинуса на вході матимуть різні діапазони значень. Маємо такі вхідні дані: швидкість поширення звуку 300 м/с, відстань в парі мікрофонів становить 30 см, тактова частота становить 44100 Гц. Після проведення обчислень на основі усіх вхідних даних, отримуємо результати, наведені в табл. 2.1.

Таблиця 2.1 — Значення кутів на основі косинусів для частоти дискретизації 44100 Hz

Зсув n	cosa	а в радіанах	а в градусах
0	0	1,570796	90
1	0,022676	1,548119	88,70066
2	0,045351	1,525429	87,40066
3	0,068027	1,502717	86,09932
4	0,090703	1,479969	84,79595
5	0,113379	1,457173	83,48988
6	0,136054	1,434319	82,1804
...	...	...	...
38	0,861678	0,532229	30,49449
39	0,884354	0,485689	27,8279
40	0,907029	0,434622	24,90198
41	0,929705	0,377185	21,61109
42	0,952381	0,309845	17,75279
43	0,975057	0,22382	12,82394
44	0,997732	0,067356	3,859228

Із табл. 2.1 видно, що для значень кута близько 90° , ми маємо різницю значень від 1 до 2 градусів, але при дедалі більшій зміні кута, різниця також зростає, і, навіть, при значенні косинуса, близькому до одиниці, різниця становить 9° , що є доволі великою. Оскільки ми досліджуємо частоту дискретизації 44100 Гц, точність є задовільною, але в табл. 2.2 результати на частоті дискретизації 8000 Гц.

Таблиця 2.2 — Значення кутів на основі косинусів для частоти дискретизації 8000 Hz

Зсув n	cos a	а в радіанах	а в градусах
0	0	1,570796	90
1	0,125	1,445468	82,81924
2	0,25	1,318116	75,52249
3	0,375	1,1864	67,97569
4	0,5	1,047198	60
5	0,625	0,895665	51,31781
6	0,75	0,722734	41,40962
7	0,875	0,505361	28,95502
8	1	0	0

Із табл.2.2 видно, що кількість даних зменшилася, як і точність розрахунків, і похибка дорівнює 7° в кращому випадку і 30° - у гіршому. Для зменшення впливу даної особливості використаної функції, необхідно позиціонувати мікрофонні пари під різними кутами, що дає можливість розрахувати дані і отримати набір достовірних і точніших результатів, а не однакових наборів даних, із мінімальними відмінностями, на розрахунок яких витрачаються апаратні та часові ресурси. Слід зазначити, що коригування для кутів є одним із джерел похибок, оскільки вони були визначені не з абсолютною точністю, а наближеною, так що фактичне положення мікрофонів у кластерах (приймачах) не може повністю збігатися з зазначеними кутами. Також слід зазначити, що при більших відстанях до джерела звуку, набори даних будуть більш подібні, оскільки абсолютний кут ставатиме менше, різниця в відстані між мікрофонами не настільки значна. Щоб збільшити дистанцію, на якій працює функція, потрібно збільшити відстань між приймачами в кластері або збільшити частоту дискретизації, але тоді відбудеться зниження швидкості оброблення даних.

2.3 Розробка оптимальної архітектури кластерів

У даній роботі було визначено та використано такі моделі кластерів приймачів: трикутна, лінійна, квадратна, дельтоїдна. Дані конфігурації показано в Додатку В із вершинами, кожна з яких має свій номер. Найбільший край (найбільша дистанція в мікрофонній парі) задається програмно, ця відстань також відображається зеленим кольором для наведених конфігурацій.

Ці конфігурації були обрані під час підготовки завдання на основі важливої вимоги конфігурації — робочий кластер повинен бути спроможним записувати звук, що надходить від джерела в зоні 360° для подальшого оброблення програмою. Ця необхідність породжує дві задачі:

- максимальне підвищення точності результируючих даних;
- збільшення відстані в парах приймачів для точнішого визначення

затримки і, отже, менший кутовий діапазон до джерела звуку.

Задача збільшення відстані не є проблемою, оскільки збільшення дистанції у парах означає точніші результати, а для отримання як можна вищої точності необхідно встановити якомога більшу відстань. Більше значення для відстані у парах дозволяє зробити затримку між сигналами на мікрофоні вищою, і, як наслідок, підвищити зсув сигналів і покращити результат, отриманий у процесі розрахунку косинуса кута, точніше визначити арккосинус розрахованого кута і звузити кутовий проміжок, в якому знаходиться джерело.

Для підвищення точності потрібно розділити діапазон 360° на сектори з певним значенням кута. Розглянемо рис. 2.5.

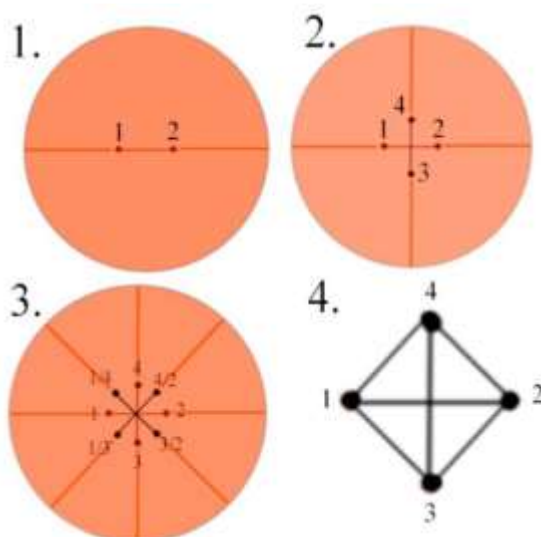


Рисунок 2.5 — Поділ діапазону на сектори за правильного розташування пар

З однією парою ми отримуємо охоплення 180° (від 0° до 180° , від 180° до 360°). Із результатів табл. 2.1 і 2.2 можна побачити, що найвища точність спостерігається в секторах поблизу 90° і 270° і зменшується, коли джерело наближається до 180° і $0^\circ/360^\circ$. Додавши перпендикулярно до існуючої ще одну пару мікрофонів, отримаємо зменшення зони покриття пари від 180° до 90° . Цей поділ видає точність пошуку кута від 1 до 2° , чого не досягти з однією парою. Крім того, кутова точність у такому разі найменша при 45° , 135° , 225° , 315° , за

постійних умов розрахунку для наведених у табл. 2.1 результатів точність становить від 2° до 3° . Однак, додаючи і правильно комбінуючи пари мікрофонів, ми формуємо 4 пари, які в ідеальних умовах (при правильному взаємному розташуванні мікрофонних пар) ділять діапазон на 8 секторів, зменшуючи сектор аналізу певної пари з 90° до 45° . Слід зазначити, отриманий результат також не є достатнім, тому що при обчисленні крос-кореляції необхідно двічі перевірити кожен мікрофонну пару, щоб знайти зміщення даних, репрезентуючих сигнал. Наприклад, на рис. 2.1 під час візуального аналізу другого сигналу матимемо зміщення на певну величину щодо першого. Якщо сигнал, що перевіряється, зсувається, а аналізований сигнал не зміщується, функція не знайде жодного результату, і результатом буде 0. Наприклад, за умови короткого сигналу, аналізований набір відображатиме синусоїду, що вже спадатиме, а на тому, що перевіряється, вона буде ще зростати.

Із цієї причини розрахунок ВКФ потрібно виконати двічі для однієї пари. При повторному аналізі першого сигналу щодо другого отримаємо, що графік другого відображає графік першого із певною затримкою. У такому випадку ВКФ також знайде подібність і дасть числовий результат, відмінний від нуля.

Оскільки така особливість збільшує розрахунок пар у 2 рази, то кількість пар для обчислення буде 12, а не 6, що ділить кутовий діапазон на 12 частин, і кожному сектору відповідатиме 30° . На рис.2.3 показано кластер, що створено шляхом поділу кутового діапазону таким чином. Подальше розроблення та імплементація таких конфігурацій до системи будуть розглянуті в наступних розділах магістерської кваліфікаційної роботи.

Із метою покращення результату проведемо додаткову оптимізацію алгоритму. Під час аналізу отриманих після роботи алгоритму даних виявлено, що навіть у найкращій конфігурації є пара, результат знайденого кута якої, відрізняється від дійсного найбільше. Для того, щоб мінімізувати результат такої пари та визначення одного дійсного кута на пару, проведено оптимізацію знаходження результуючого кута шляхом визначення мінімальної кутової

дисперсії.

Нехай, маємо 6 пар. Для кожної пари алгоритм знаходить по 2 кути, а отже існує 64, чи 2^6 комбінацій кутів кожної пари. Мінімальну дисперсію можна знайти шляхом розрахунку із усіх можливих комбінацій пар, паралельно визначаючи правильні кути, на основі розрахунку дисперсії.

Після проведених розрахунків для оптимізації виведення даних на область моделювання, пара, що дає найгірший результат, відкидається, а з правильних значень усіх інших пар вираховується середнє арифметичне, яке і застосовується для візуального виведення даних.

Для подальшого відображення знайденого кута до джерела, використаємо формулу для визначення точки на колі. За її допомогою вирахуємо X та Y координати точки під необхідним кутом. Таким чином, проведемо лінію із центру кола (приймача) до точки на колі і отримаємо напрям до джерела (рис.2.3). В ідеальних умовах, при абсолютно точних умовах, пряма має проходити і через центр джерела звуку, але через похибки в процесі роботи алгоритму, не досконало точних даних, розрахунків геометричних та тригонометричних функцій, результат спотворюється і лінія може відхилятися від джерела звуку на певне значення.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛІ СИСТЕМИ ЛОКАЛІЗАЦІЇ ЗВУКОВИХ ДЖЕРЕЛ

3.1. Розробка програмного забезпечення моделі системи локалізації звукових джерел

Перед програмною реалізацією запропонованої в магістерській роботі системи аудіолокації, спочатку варто обґрунтувати архітектуру додатку, його структуру. Це уможливить мінімізацію ризиків, визначить процес її подальшого розроблення, модифікацію, інтеграцію з бізнес-процесами. Проте, існує кілька видів ПЗ, кожен з яких має свою специфіку роботи. Тож, щоб мінімізувати подальші фінансові втрати на модифікацію, доцільно визначити, до якого класу належить продукт, що створюється. Згідно зі схемою з рис. 3.1, існують такі основні види ПЗ: інструментальне, системне та прикладне. Потрібно визначити, яким саме варто реалізовувати систему аудіолокації.



Рисунок 3.1 — Класифікація ПЗ

Прикладне забезпечення - це вид ПЗ, що використовується користувачем під час виконання завдання для вирішення певної задачі. Таке ПЗ називають також застосунком. Воно є найбільш поширеними на ринку. Як правило, застосунки працюють лише за умови наявності системного ПЗ, яке слугує фундаментом для стабільної роботи застосунків. Через поширеність даного виду ПЗ з'явилася додаткова класифікація за призначенням. Так, вони є спеціалізовані

та загального призначення. З назви зрозуміло, що загальні ПЗ – ті, які можуть використовуватися в різних сферах, зокрема, текстові редактори, додатки соц. мереж, коли ж спеціалізовані виконують певну конкретну функцію і використовуються за специфічних умов. Наприклад, засоби для редагування мультимедійних файлів, що розповсюджуються безкоштовно, можуть бути використані для ознайомлення із функціоналом або навчання, коли повна версія містить багато інструментів, які знадобляться лише спеціалісту саме для високоякісного оброблення мультимедійних файлів. ПЗ, що було розроблено під час виконання даної роботи належить саме до цієї категорії, оскільки воно є спеціалізованим і йому потрібно запускатися на пристроях із підготованими умовами, які може забезпечити певне системне ПЗ. До інструментального програмного забезпечення належать спеціалізовані системи, які містять кілька програм, метою яких є побудова машинного коду. Даним видом ПЗ користуються програмісти для написання програм. Для ПЗ, що розроблялося в даній роботі, також було використано інструментальне ПЗ - Visual Studio. Дана система включає більшість важливих для розробки засобів, наприклад бібліотеки, редактор коду та графічного інтерфейсу, компілятор, лінкер та інші корисні інструменти. Зазвичай таке ПЗ обирається на основі декількох критеріїв, а саме: вимоги та політика компанії, інтеграція засобів розробки між собою, простота освоєння, ефективність ПЗ, вибір середовища.

Системне програмне забезпечення – вид програм, що використовуються як основа для запуску інших застосунків, а також мають доступ до апаратної частини і організовують її оптимальну та надійну роботу і взаємодію апаратної частини, та високорівневих застосунків [41, 42]. Найяскравішим представником даного виду ПЗ є операційні системи (ОС).

Зазвичай саме ОС виконує функцію керування всіма процесами, запущеними на комп'ютері, а також їх прискоренням. Відомими ОС є Linux, Windows, MacOS. Кожна з них має свої особливості. Наприклад, Windows вважається найбільш зручною за рахунок графічного інтерфейсу. Проте, через

монолітність та надлишкову навантаженість, її потрібно оптимізувати, налагодити взаємодію модулів та трапляються збої, що не дають всій системі нормально функціонувати і виникає так званий синій екран. Linux навпаки, модульна і легка, швидка, спеціалісти її цінують, проте відсутній графічний інтерфейс і всі налаштування в ОС зазвичай виконуються через консоль, що ускладнює знайомство із даною ОС. Що стосується MacOS, то вона є гібридом попередніх двох, що виправив мінуси попередніх двох, проте і яскраво виражених сильних аспектів у неї немає.

Основними характеристиками системного ПЗ зазвичай є:

- висока швидкодія за рахунок оптимізації та забезпечення достатньої швидкості роботи ПЗ вищого рівня;
- великий спектр можливостей по налаштуванню, проте, задля повного доступу до них, потрібно маніпулювати командами мови програмування, а не графічним інтерфейсом;
- даний вид ПЗ має бути написаний на мові низького рівня, щоб центральний процесор міг обробляти і виконувати команди, що надходять від ОС;
- системне ПЗ працює безпосередньо із апаратним забезпеченням;
- має бути універсальним та слугувати інтерфейсом між програмами та обладнанням, на якому воно встановлено.

Системне ПЗ має також підтримувати свої застосунки [43]. Розроблене програмне забезпечення належатиме саме до прикладного спеціалізованого, оскільки буде використовуватися лише спеціалістами певної галузі для виконання специфічних завдань, створене для обробки масивів спеціалізованої інформації, матиме графічний інтерфейс для взаємодії із користувачем. З урахуванням вузької спеціалізації розробленого ПЗ, для покращення роботи користувачів зі створеною системою особливу увагу буде приділено користувацькому інтерфейсу та створенню бази знань, що використовуватиме програма під час роботи. Оскільки на меті у ПЗ, що розробляється, знаходити

звук, то в програмі має бути представлено приймачі, джерела звуку. У програмі здійснюватимуться також розрахунки, які не стосуються ні мікрофонів, ні джерела, оскільки в реальності вони лише приймають або генерують звук, відповідно. Тому створимо класи, що відображатимуть ці розрахунки. Перший – відображатиме розроблений метод для знаходження кута, а інший буде використано для правильного і наближеного до реального процесу затримання даних, коли ввімкнені мікрофони, відбувся спалах від пострілу, а звук ще не дійшов, а коли дійде, то на різні мікрофони він потрапить із різницею в часі. Для відображення розрахунків потрібно створити певний інтерфейс засобами якого організовуватиметься взаємодія із програмою. Він буде виводити результат у певну форму. Створимо додатковий клас для дублювання даних у файл і на консоль у текстовому вигляді. Для відображення у користувацькому інтерфейсі джерела звуку та приймача потрібно також додати певні сутності, зокрема, додаткові класи, основна функція яких полягатиме у виведенні певного об'єкта на користувацький інтерфейс. Отож, структуру ПЗ наведено на рис.3.2.

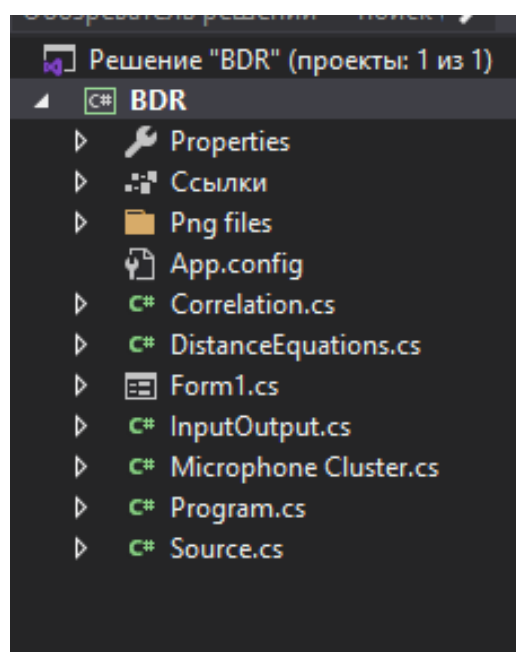


Рисунок 3.2 — Структура розробленого ПЗ

Трикутник запрограмовано як рівносторонній, що має мікрофон на перетині медіан, бісектрис і висот. Ця точка також є центром даного кластера, який використовується для розрахунку положення вершин трикутника. Дана конфігурація є однією з кращих, оскільки кожна окрема пара розташована під кутом до будь-якої іншої. Оскільки умова місцезнаходження мікрофонних пар виконується, то для кожної пари отримуємо сектор в 30° . Однак, оскільки центральний мікрофон розташований по центру конфігурації, що є недоліком, то відстань у парі потрібно збільшувати для більшої точності.

Програмний код для розрахунку відстаней у конфігурації показано на рис. 3.3. Лінія є найгіршим варіантом для обчислень, оскільки будь-яка мікрофонна пара знаходиться паралельно (на одній лінії) щодо іншої пари в кластері з однаковим кутом до джерела. За умови, що джерело звуку знаходиться на достатній відстані, отримаємо майже ідентичні, якщо не враховувати завад, набори даних, після оброблення яких матимемо один і той самий результат.

```
else if (mc.isTriangle) {
    dist = new double[] { biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro,
        biggestDistanceBetweenMicro,
        biggestDistanceBetweenMicro
    }
}
```

Рисунок 3.3 — Програмна реалізація розрахунку відстаней між приймачами в парах трикутної конфігурації між мікрофонами

На рис. 3.4 — дистанції від джерела до кожної вершини з метою коректного моделювання процесу пошуку джерела звуку.

```

else if (mc.isTriangle) {
    distX[0] = (xDest - xSource) * SCALE;
    distY[0] = (yDest - ySource) * SCALE;
    distX[1] = (xDest - xSource) * SCALE;
    distY[1] = (yDest - ySource) * SCALE - len / Math.Sqrt(3);
    distX[2] = (xDest - xSource) * SCALE - len / 2;
    distY[2] = (yDest - ySource) * SCALE + len / 2 / Math.Sqrt(3);
    distX[3] = (xDest - xSource) * SCALE + len / 2;
    distY[3] = (yDest - ySource) * SCALE + len / 2 / Math.Sqrt(3);
}

```

Рисунок 3.4 — Програмна реалізація розрахунку розташування вершин щодо джерела в трикутній конфігурації

Щоб зменшити негативний вплив розміщення мікрофонів на конфігурацію, їх було розміщено на лінії з різними дистанціями між мікрофонами, що утворило 6 пар (із різними відстанями між мікрофонами), які не повторювали-ся. На рис.3.5 розрахунок відстаней між вершинами.

```

dist = new double[] {
    (biggestDistanceBetweenMicro/3) - ((biggestDistanceBetweenMicro/3)/4),
    (biggestDistanceBetweenMicro/3) -
    ((biggestDistanceBetweenMicro/3)/4) + biggestDistanceBetweenMicro/3,
    biggestDistanceBetweenMicro,
    biggestDistanceBetweenMicro/3,
    (biggestDistanceBetweenMicro/3) +
    ((biggestDistanceBetweenMicro/3)/4) + biggestDistanceBetweenMicro/3,
    (biggestDistanceBetweenMicro/3) +
    ((biggestDistanceBetweenMicro/3)/4)
};

```

Рисунок 3.5 — Програмна реалізація розрахунку відстаней між вершинами в лінійній конфігурації

На рис. 3.6 представлено програмний код для обчислення відстаней від джерела звуку до вершин.

```
if (mc.isLine) {
    distX[0] = (xDest - xSource) * SCALE - len/2;
    distY[0] = (yDest - ySource) * SCALE;
    distX[1] = (xDest - xSource) * SCALE - len / 2 + (len / 3 - len / 6);
    distY[1] = (yDest - ySource) * SCALE;
    distX[2] = (xDest - xSource) * SCALE + len / 2 - (len / 3 - len / 6);
    distY[2] = (yDest - ySource) * SCALE;
    distX[3] = (xDest - xSource) * SCALE + len / 2;
    distY[3] = (yDest - ySource) * SCALE;
}
```

Рисунок 3.6 — Програмна реалізація розрахунку розташування вершин щодо джерела у конфігурації

Із можливих 6 пар всі даватимуть різні дані, але із сектором 180° , що унеможливилює достатньо точне визначення напрямку до шуканого джерела звуку, коли воно близьке до 180° або близько до $0^\circ/360^\circ$. Розрахунок позицій приймачів у конфігурації було проведено із центру відрізка. Максимальна відстань у даній конфігурації — це відстань між крайнім лівим та крайнім правим мікрофонами.

Квадратна конфігурація через свою структуру по результативності схожа на щось між трикутною та лінійною, тому що є паралельні пари, в яких однакова відстань, кути до джерела, що зменшує кількість потенційно корисних пар з 6 до 4, а тому отримуємо кутові сектори по 45° . Середина конфігурації — точка, в якій перетинаються діагоналі. Від неї обчислюються позиції всіх вершин. На рис.3.7 показано розрахунок відстаней між вершинами.

```

dist = new double[] { biggestDistanceBetweenMicro,
    biggestDistanceBetweenMicro * Math.Sqrt(2),
    biggestDistanceBetweenMicro, biggestDistanceBetweenMicro,
    biggestDistanceBetweenMicro * Math.Sqrt(2),
    biggestDistanceBetweenMicro };

```

Рисунок 3.7 — Програмна реалізація розрахунку відстаней в кожній парі мікрофонів квадратної конфігурації

На рис.3.8 наведено обчислення відстаней до джерела звуку

```

else if (mc.isSquare) {
    distX[0] = (xDest - xSource) * SCALE - len / 2;
    distY[0] = (yDest - ySource) * SCALE - len / 2;
    distX[1] = (xDest - xSource) * SCALE + len / 2;
    distY[1] = (yDest - ySource) * SCALE - len / 2;
    distX[2] = (xDest - xSource) * SCALE + len / 2;
    distY[2] = (yDest - ySource) * SCALE + len / 2;
    distX[3] = (xDest - xSource) * SCALE - len / 2;
    distY[3] = (yDest - ySource) * SCALE + len / 2;
}

```

Рисунок 3.8 — Програмна реалізація розрахунку дистанцій від джерела до кожного мікрофона у квадратній конфігурації

Однією серед кращих є дельтоїдна конфігурація. Вона дає найбільш точні результати, оскільки кожна мікрофонна пара, відносно будь-якої іншої пари у конфігурації не є паралельною і знаходиться під певним кутом. Розташування вершин схоже як у трикутнику, і кутовий сектор кожної пари в якому становить

30°, але на відміну від якого, мікрофон розташований не в центрі, а винесений назовні, що дозволило збільшити в цій конфігурації відстань у парах мікрофонів, що, у свою чергу, дозволяє отримувати більш точні дані, порівняно з трикутником, за рахунок збільшеної відстані. Середня точка — аналогічна центральній точці із трикутника, найбільша дистанція в мікрофонній парі така ж, як і у трикутника. Проте, оскільки центральний мікрофон не в центрі трикутника, то потрібно провести перерозрахунок для цієї вершини і дистанцій в парах конфігурації. Програмну реалізацію розрахунку відстаней між вершинами показано на рис.3.9.

```
if (mc.isFlyingSnake)
{
    dist = new double[] { 2*biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro / Math.Sqrt(3),
        biggestDistanceBetweenMicro, biggestDistanceBetweenMicro,
        biggestDistanceBetweenMicro };
}
```

Рисунок 3.9 — Програмна реалізація розрахунку відстаней в кожній парі мікрофонів дельтоїдної конфігурації

На рисунку 3.10 наведено розрахунок відстаней до джерела звуку.

```
else if (mc.isFlyingSnake) {
    distX[0] = (xDest - xSource) * SCALE;
    distY[0] = (yDest - ySource) * SCALE + len / Math.Sqrt(3);
    distX[1] = (xDest - xSource) * SCALE;
    distY[1] = (yDest - ySource) * SCALE - len / Math.Sqrt(3);
    distX[2] = (xDest - xSource) * SCALE - len / 2;
    distY[2] = (yDest - ySource) * SCALE + len / 2 / Math.Sqrt(3);
    distX[3] = (xDest - xSource) * SCALE + len / 2;
    distY[3] = (yDest - ySource) * SCALE + len / 2 / Math.Sqrt(3);
}
```

Рисунок 3.10 — Програмна реалізація розрахунку дистанцій від джерела до кожного мікрофона в дельтоїдній конфігурації

3.2 Створення та реалізація інтерфейсу користувача

Інтерфейс користувача або UI — інтерфейс, що дозволяє спростити комунікацію між програмою та користувачем. Зазвичай, взаємодія відбувається за допомогою спеціальних елементів. З боку користувача такими є кнопки, перемикачі, списки, поля введення та інше для введення інформації. З боку програми відбувається виведення інформації у певних полях, як от діалогові вікна з текстом, картинки, поля із інформацією, графіка і т. ін.

Перш за все, потрібно розробити функціонал програми, що зумовлює певні елементи користувацького інтерфейсу та запрограмувати їх взаємодію із людиною та застосунком. Програма має надавати змогу задавати вхідні дані:

- значення для масштабування області моделювання;
- значення для максимальної відстані в клайстері приймачів;
- значення частоти дискретизації для правильної роботи алгоритмів програми;
- тип клайстера, який буде розташований на області для моделювання;
- відображення результатів на консолі та на графічній області;
- графічна область, яка відображатиме елементи і візуалізуватиме результати обчислень у режимі реального часу;
- обрати вид пристрою для генерації чи захоплення сигналу;
- запуск розрахунків;
- збереження результатів у файл;
- зміна розміру області моделювання.

Виділимо відповідний функціонал програми щодо інформації, яку надаватиме користувач засобами користувацького інтерфейсу.

- запам'ятовувати частоту дискретизації;
- змінювати вказівник при виборі пристрою генерації чи отримування сигналу;
- запам'ятовувати масштаб;
- відображати масштаб області для моделювання;

- перемальовувати область моделювання при внесенні змін;
- запам'ятовувати обраний тип кластера;
- запам'ятовувати найбільшу дистанцію в кластері;
- відобразити додаткові дані на області для покращення користування програмою;
- повідомити користувача про певну помилку, якщо вона виникає;
- продублювати отримані дані в кілька джерел виводу;
- почати обчислення на основі вхідних даних та надати їх користувачу.

Розробка всього функціоналу і правильна його взаємодія із користувацьким інтерфейсом дозволяє створити програму, що буде виконувати всі поставлені задачі та зменшить час на опанування нею оператором.

Почнемо розробку із загального макета користувацького інтерфейсу. Він представляє собою звичайне вікно прикладної програми ОС Windows. Саме це вікно і буде шаблоном для майбутньої області моделювання.

Для реалізації елементів, що дозволять задавати певні вхідні дані використаємо елемент панель. На ній розташуємо заготовки під майбутні елементи керування, а також кнопку, яка відповідатиме за запуск розрахунків. Створений шаблон можна побачити на рис. 3.11.

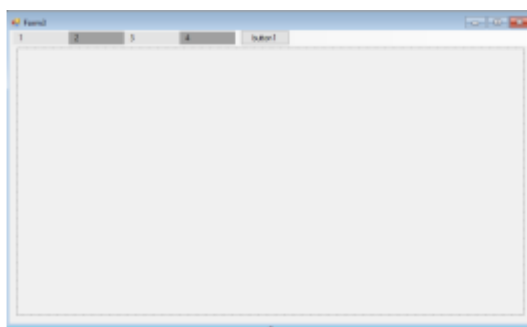


Рисунок 3.11 — Вигляд заготовки

Для налаштування частот дискретизації, яких є кілька, потрібно відобразити всі елементи, не загромождаючи інтерфейс користувача, який має

бути простим і мінімалістичним. Виконати дану задачу найкраще можуть випадаючі списки. Їх особливість полягає в тому, що список виглядає як звичайна кнопка, не займаючи багато місця, а подія на яку він запрограмований, розширюється і показує весь вміст. Також додамо інтерфейс, що дозволить обирати певну конфігурацію кластера, для цього створимо додаткові кнопки із місцями для ілюстрацій та поле введення, що призначене для налаштування обраної конфігурації. Всі додані елементи є на рис.3.12.



Рисунок 3.12 — Вигляд заготовки після внесення змін

Для більшої інтуїтивності інтерфейсу програми додамо деякі елементи. Так, перший список — «Frequency», тут можна обрати частоту дискретизації, Другий список — «Device», реалізований як схематичне графічне відображення конфігурацій із підписами на полі для моделювання. Третій список, що дозволяє обрати масштаб, який відповідає одному пікселю, називається «Scale». Останній список, який містить відображає джерело, має назву «Source». На додачу до цього переназвемо і пункти списків, на моделюючу область накладемо картинку карти, а в місцях для графічного представлення конфігурацій додамо рисунки, що їм відповідають. Результат відображено в Додатку Г. Наступним кроком у розробленні ПЗ є візуальні підказки під час вибору та розміщення конфігурації на полі моделювання. Отже, було запрограмовано такий механізм. Графічно відображається підказка, що означає зміну вказівника під час наведення на поле моделювання та вже обраної конфігурації, тобто, коли програма готова до

використання конфігурації. Після розташування пристроїв на області, курсор змінюватиметься назад на стандартний, що означає можливість вибору іншого пристрою, за необхідності, та розміщення його на області. Обрану конфігурацію клайстера чи джерела можна змінити, доки елемент не було розташовано. Можна використовувати й інші елементи керування програмою. Зміна вказівника не обмежує користувача, а лише символізує, що є певний об'єкт, який можна використати. Роботу даної підказки відображено на рис. 3.13.

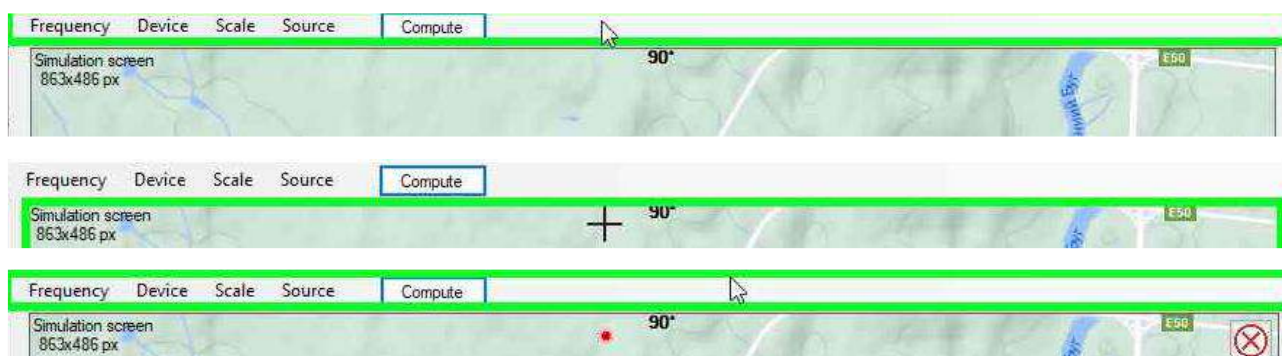


Рисунок 3.13 — Відображення підказки в інтерфейсі для моделювання

Виведення результатів можна реалізувати кількома шляхами: графічне відображення результатів або виведення їх у вигляді текстових даних. Текстовий варіант поділяється на консольний і файловий вивід. Розглянемо детально всі типи виведення.

За файлового варіанту отримані дані надходять до файлу, що дозволяє виводити великий обсяг даних, зберігати його у хронологічній послідовності та використовувати у подальшому для аналізу, модифікації алгоритмів. Недоліком такого методу є те, що дані неможливо візуалізувати. Виведені дані у файлі будуть незрозумілі для непідготовлених операторів. Операція виведення даних у файл є ресурсозатратною та часозатратною, що стає особливо обтяжливим при постійному виведенні великих масивів даних. Неможливо проглянути вміст файлу до завершення роботи програми, а отже, і впевнитися, що програма заносить коректні дані.

Консольний вивід схожий на вивід у файл, проте є швидшим і менш ресурсозатратним, дозволяє візуалізувати отримані дані у вигляді тексту. Незважаючи на це, він також не позбавлений недоліків, наведемо основні. Складність для непідготовленого користувача. При виведенні великої кількості даних можлива ситуація, коли консоль швидко оновлюється і спеціаліст не встигатиме обробляти дані. Дані доступні лише під час роботи програми, після її закриття всі дані втрачаються.

Останній вид виведення даних — графічний. Він протилежний за своїми особливостями та недоліками до текстових. Найбільш значущою перевагою можна назвати просту і зрозумілу візуалізацію результатів, за умови, що він був оптимізований і мінімізований. Проте, такий вид виведення не доступний після припинення роботи програми, його неможливо використовувати для аналізу. Отже, для програмної реалізації звукометричної системи було скомбіновано консольний та графічний варіанти виведення даних, що забезпечило наглядність, можливість аналізу в подальшому, при цьому надто швидке оновлення даних чи зниження швидкодії нівелюються. Наявним недоліком є відображення даних на консолі для непідготованих користувачів та недоступність даних після закінчення виконання застосунку. Другий недолік можна нівелювати шляхом ручного копіювання даних із консолі у файл, під час виконання програми. Таким чином, отримуємо програму, що фактично позбавлена вище описаних недоліків.

Результат комбінованого виведення наведено на рис. 3.14, де видно текстову частину

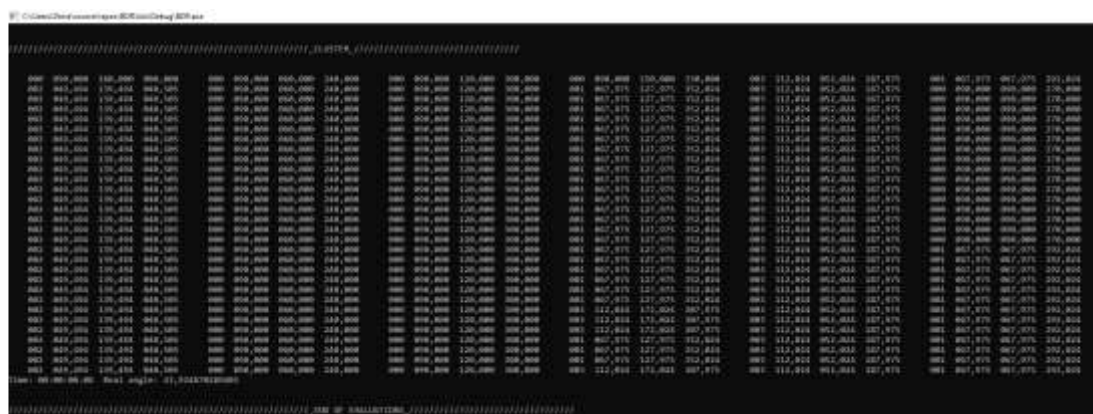


Рисунок 3.14 — Виведення результату роботи програми на консоль

На рис.3.15 наведено графічну частину.

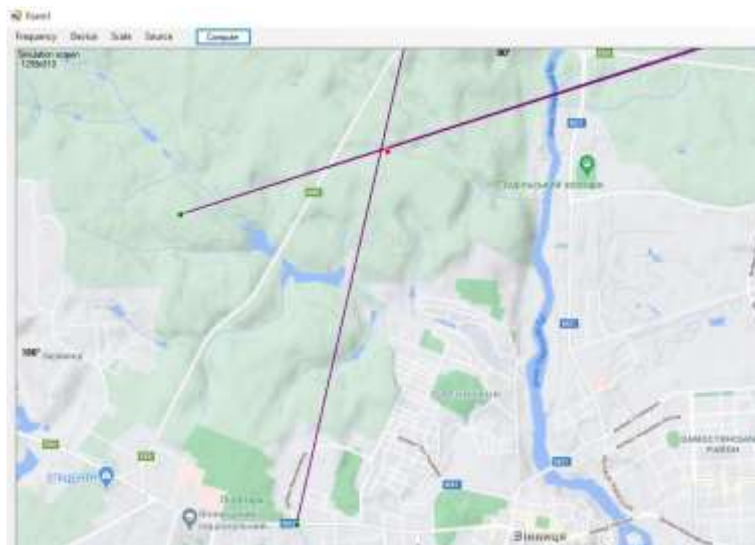


Рисунок 3.15 — Виведення результату роботи програми на область моделювання

3.3 Тестування програмної моделі системи

Існує багато типів тестів, найважливішими серед них є такі, що базуються на: ступені автоматизації, ступені тестової готовності, цілях, об'єктах, ізоляції складових, періоді виконання, системних знаннях, показнику сценарної позитивності. Перевірку правильності роботи алгоритмів ПЗ можна виконати різними типами тестів. Було використане функціональне тестування вручну, що зумовлено детальним розглядом ланцюга виконання перевірки функціональності. Так, зазвичай, для певних функцій створюються відповідні документи, далі вони і «фічі» передаються тестувальнику, який перевіряє цей функціонал. Після тестування, тестер створює документ, що містить в собі інформацію про певні недоліки, та як їх відтворити. Даний документ досліджує розробник, виправляє, що потрібно і знову формує документ. Таких ітерацій може бути достатньо багато, що є часовитратним. Процес тестування спричиняє додаткові затримки в тестуванні продукту. Набагато простіше розробнику знайти недолік, виправити його та перевірити виправлення. Даний варіант дозволяє економити багато часу за рахунок відсутності комунікацій з іншими членами команди та складання документації, особливо, якщо крім розробника та

тестувальника, нікого в команді немає. Таким чином, в даній роботі також використовувався цей тип тестування, оскільки він виконується після розробки застосунку та дозволяє перевірити функціонал, виявити помилки та виправити їх наприкінці циклу розробки. Розглянемо інший вид — модульні тести. Цей тип було б доцільно застосувати, оскільки він може виконуватися в будь-який момент циклу розробки, дані тести застосовуються системою під час запуску проєкту, для перевірки правильності роботи модулів програми, тим самим скорочуючи час тестування. Під час використання даного типу тестів, створюються певні фікстури (початкові умови) для перевірки поведінки модулів у різних ситуаціях, незалежно від інших модулів. Він також дозволяє швидко перевірити, чи вплинули певні зміни в коді на поведінку модуля, тим самим, можна відстежити і запобігти виникненню похибок в роботі програми та спростити процес модернізації застосунку. Щоб використовувати даний тип тестування, необхідно мати чітку та незмінну структуру додатку, визначену та незмінну логіку застосунку та документацію, в якій вказано функціонал програми [44, 45]. Оскільки тестування проводиться під час розроблення, то і створення тестів відбувається тоді ж, що займає час. Такий тип використовують великі компанії, які створюють великі системи із великою кількістю коду, поламки в них можуть нести репутаційні втрати. Для малих систем, як такої, що розробляється в магістерській роботі, ці тести не прийнято використовувати.

Протестуємо змогу ПЗ надавати можливість користувачам обирати частоту дискретизації, масштаб, потрібну конфігурацію приймача, джерело звуку, що дозволить розташувати їх на області для моделювання, встановити напрям та відобразить його, продублює ці дані в текстовому вигляді в консоль. Для цього виконаємо таку послідовність дій:

- 1) встановимо частоту дискретизації 441000 Гц через меню з значеннями частот;
- 2) встановимо значення 1:1 для масштабу через аналогічне меню;
- 3) встановимо бажану конфігурацію у меню «Device», задамо значення найбільшої відстані 0,5 м;

- 4) розташуємо конфігурацію на області моделювання;
- 5) повторимо крок 3, 4, обравши іншу конфігурацію;
- 6) розташуємо джерело звуку після його вибору в пункті меню «Source»;
- 7) повторимо крок 6 та розташуємо ще одне джерело звуку;
- 8) розрахуємо результат за допомогою кнопки «Compute»;
- 9) змінимо розміри сторін вікна програми, перетягуючи відповідну сторону.

Виконання перших двох кроків відображено на рис. 3.16.

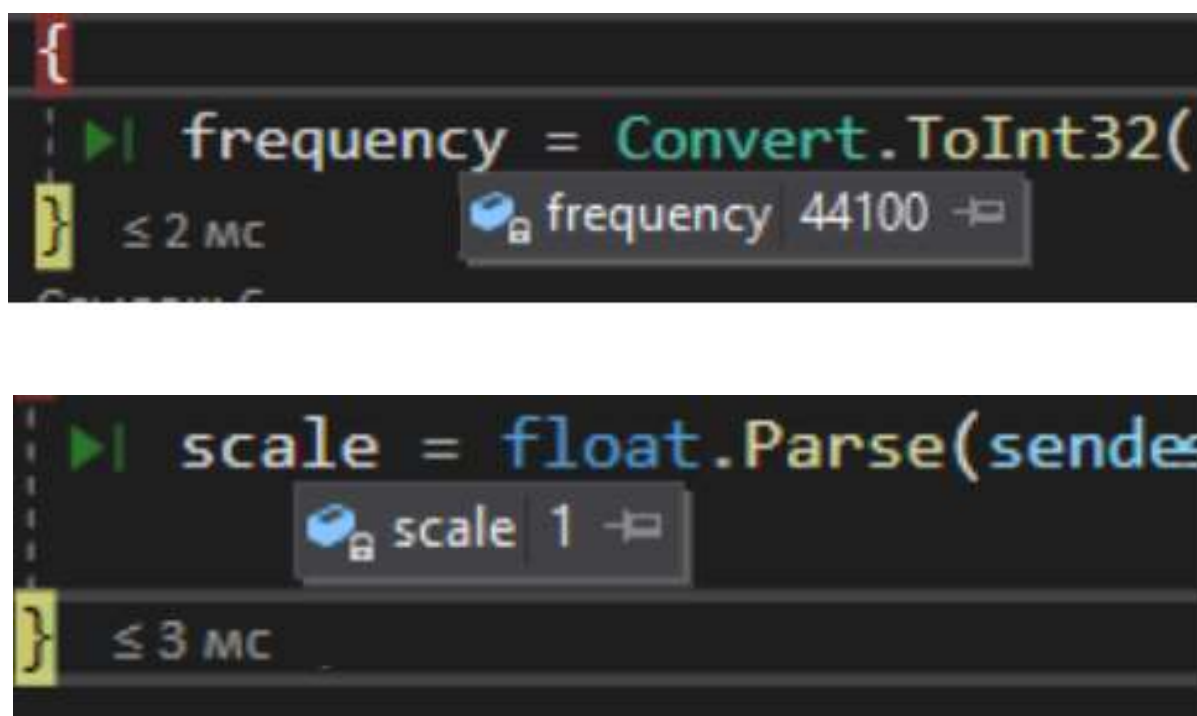


Рисунок 3.16 - Присвоєні дані після 1 та 2 кроків

Після виконання кроків 3, 4, 5 відображається меню, в якому можна встановити тип конфігурації кластера та дистанції. Після натиску кнопки, що встановлює ці дані меню, вона зникає та змінює вказівник миші із стрілки на хрестик. Результати кроків 3, 4, 5 відображаються в області моделювання. Ледь помітні дві точки, зсунуті відносно центру в лівий нижній кут, на рис. 3.17. Результат роботи кроків 6 та 7 являє собою точки, графічне представлення джерел звуку. Після виконання 8 кроку дані виводяться на графічну область та

дублюються на консолі.

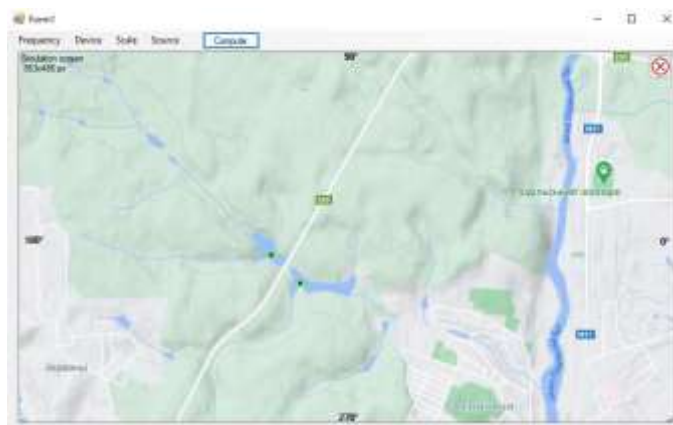


Рисунок 3.17 — Результати кроків 4 та 6

Після виконання кроку 9, з'являється новий розмір вікна та області моделювання в лівому верхньому куті (рис. 3.18). Зроблено примітки на самому рис. 3.18 для подальшого розуміння.

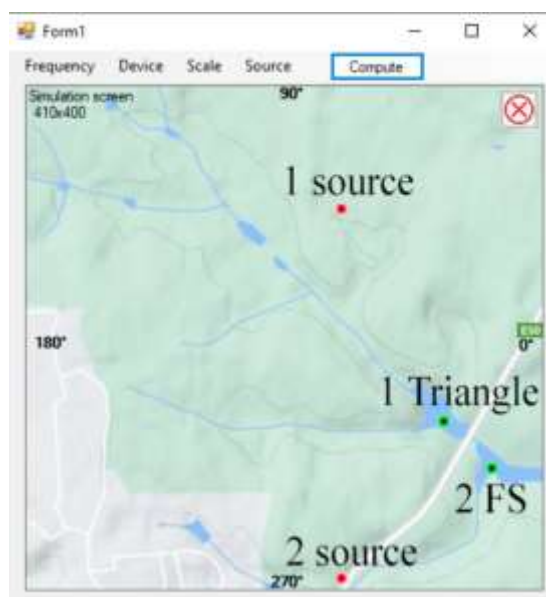


Рисунок 3.18 — Результат виконання кроку 10

Проаналізувавши результат виконання програми, можна підсумувати, що робота застосунку відповідає висунутим до нього вимогам, оскільки спілкування між користувачем і додатком засобами графічного інтерфейсу відбувається успішно.

ВИСНОВОК

В роботі було проведено аналіз відомих на ринку звукометричних систем. Такими вважаються системи «SWATS» та «Ears» від QinetiQ, «Boomerang» та її модифікації від Raytheon, «Pearl» від Acoem-Metravib та «Pilarw», AVS датчики та розроблену на їх базі систему «CASTLE» від Microflown Avisa та «Sniper Egg» від Ultra Electronics. На основі аналітичних даних було виявлено їх сильні та слабкі сторони, які погіршують можливість для модифікації, налаштування цих систем та догляду за ними, а саме: висока ціна, монолітність, потреба в адаптивності, складність у використанні.

На основі результату опрацювання аналітичних даних, що стосуються існуючих звукометричних систем, а також аналізу теоретичних та практичних результатів наукових праць їх дослідників звукометрії, було визначено можливість модифікації систем на ринку, шляхом усунення основних недоліків цих систем, завдяки покращенню алгоритму знаходження напрямку до джерела звуку. Виходячи із специфіки обробки оцифрованих сигналів, проведено його модифікацію за допомогою використання взаємкореляційної функції, а також подальшого розрахунку кута з використанням функцій косинуса та арккосинуса. Для покращення результатів, прийнято рішення про розробку різних конфігурацій кластерів приймачів, що дозволять розбити кутовий діапазон між парами приймачів у кластері. Так кожна пара буде мати сектор кутів, де визначення результату функції арккосинуса буде найточнішим. В результаті розробки конфігурацій в теорії отримали підвищення точності визначення кута, проте на практиці цього виявилось недостатньо, оскільки в кожній конфігурації потрібно визначити справжній кут до джерела та для кожної конфігурації є найбільш несприятливий результат із набору результатів, який впливає на розрахунок кута. Для вирішення даних проблем було використано статистичну обробку даних, яка дозволила визначати, який саме кут відповідає шуканому, а з правильних результатів видалявся той, що мав найбільшу похибку при обрахунках.

Наступною задачею була розробка ПЗ, що дозволить відтворювати процес знаходження напряму від програмної моделі звукометричної системи до джерела звуку. Задля цього було визначено тип програм, до якого буде відноситись розроблена програма. Проведено розробку структури майбутнього застосунку та виконано розробку логіки основних логічних елементів програми. Створення програми відбувалось на ОС Windows у середовищі розробки Vindows Studio, а також із дотриманням парадигм об'єктно орієнтованого програмування. Після впровадження логіки основних структурних елементів, було проведено розробку інтерфейсу користувача, та додано певні елементи для інтерфейсу користувача, що дозволять покращити досвід користування програмою. Оскільки процес визначення джерела по сигналу вимагає цього самого сигналу, то було використано оцифрований запис справжнього звуку для моделювання процесу. Було також реалізовано вивід програмою після виконання розрахунків, графічно на область моделювання, а також текстово в консоль для можливості візуалізації і аналізу отриманих даних.

При проведенні тестування було перевірено функціонал інтерфейсу та результатів, які було отримано в результаті перевірки. Визначено, що інтерфейс виконує всі функції, які від нього вимагаються, а похибка результатів роботи алгоритму у межах допустимої. Зроблено висновок, що розроблена система позбавлена основних недоліків існуючих систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мах Е. Про швидкість поширення звуку, що збуджується різкими пострілами. Відень. 1888. Т. 11. № 98. С. 1045–1052.
2. Густав Фехнер – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Густав_Фехнер
3. Jeffress L. A. A place theory of sound localization. *Journal of Comparative and Physiological Psychology*. 1948. Vol. 41, no. 1. P. 35–39. URL: <https://doi.org/10.1037/h0061495>
4. Rayleigh J. W. S. 3. B. The theory of sound. 2nd ed. Macmillan, 1896. Vol. 1 : Volume 1.
5. Rayleigh J. W. S. 3. B. The theory of sound. 2nd ed. Macmillan, 1897. Vol. 2 : Volume 2.
6. Stevens S. S., Newman E. B. The Localization of Actual Sources of Sound. *The American Journal of Psychology*. 1936. Vol. 48, no. 2. P. 297. URL: <https://doi.org/10.2307/1415748>
7. Strutt J. W. On our perception of the direction of a source of sound. *Proceedings of the London Mathematical Society*. 1876. S1-4, no. 1. P. 75–84. URL: <https://doi.org/10.1112/plms/s1-4.1.93>
8. Localization of multiple sound sources with two microphones / C. Liu et al. *The Journal of the Acoustical Society of America*. 2000. Vol. 108, no. 4. P. 1888–1905. URL: <https://doi.org/10.1121/1.1290516>
9. Бугайов М. В. Алгоритм виявлення акустичних сигналів безпілотних літальних апаратів. *Вісник ЖДТУ*. 2016. Т. 74, № 3. С. 46–53. URL: <http://vtn.ztu.edu.ua/article/view/65584>
10. Кочан Р. В., Трембач Б. Р., Трембач Р. Б. Концепція розподіленої автоматичної системи звукової артилерійської розвідки на базі стільникового зв'язку. *Вісник Хмельницького Національного Університету*. 2016. Т. 2. С. 101–104. [http://journals.khnu.km.ua/vestnik/pdf/tech/pdfbase/2016/2016_2/\(35\)%202016-2-t.pdf](http://journals.khnu.km.ua/vestnik/pdf/tech/pdfbase/2016/2016_2/(35)%202016-2-t.pdf)

11. Кочан Р. В., Кочан О. В., Трембач Б. Р. Методична похибка пеленгування цілі системою звукової артилерійської розвідки. *Вимірювальна техніка та метрологія*. 2019. Т. 80, № 3. С. 10–14. URL: <https://ena.lpnu.ua/items/fa4d4479-7a42-4987-904c-6f49084cc264>
12. A. Dehghan Firoozabadi et al. 3D Multiple Sound Source Localization by Proposed T-Shaped Circular Distributed Microphone Arrays in Combination with GEVD and Adaptive GCC-PHAT/ML Algorithms . *Sensors*. 2022. Vol. 22, no. 3. P. 1011. URL: <https://doi.org/10.3390/s22031011>
13. Jia M et al. Multiple Sound Sources Localization with Frame-by-Frame Component Removal of Statistically Dominant Source. *Sensors (Basel)*. 2018. Vol. 11, no. 18. P. 36—43. URL: <https://doi.org/10.3390/s18113613>
14. M. Hahmann et al. Sound source localization using multiple ad hoc distributed microphone arrays . *JASA Express Lett.* 2022. Vol. 7, no. 2. P. 22–28. URL: <https://doi.org/10.1121/10.0011811>
15. F. Katzberg et al. Measurement of sound fields using moving microphones. *arXiv.org e-Print archive*. URL: <https://arxiv.org/pdf/1609.09390v1.pdf>
16. X. Li et al . Reverberant Sound Localization with a Robot Head Based on Direct-Path Relative Transfer Function. *https://arxiv.org*. URL: <https://arxiv.org/pdf/2012.03574v1.pdf>
17. H. Xing et al .A fusion algorithm of passive sound source localization based on the two-plane four-element cross array. *Review of Scientific Instruments*. 2020. Vol. 91, no. 3. P. 23–33. URL: <https://doi.org/10.1063/1.5080761>
18. L. Zhang et al. Sound Source Localization Using Non-Conformal Surface Sound Field Transformation Based on Spherical Harmonic Wave Decomposition. *Sensors*. 2017. Vol. 17, no. 5. P. 1087. URL: <https://doi.org/10.3390/s17051087>.
19. Азарова А., Щур Д. Побудова автоматизованої системи аудіолокації загроз. *Інформаційні технології та комп'ютерна інженерія*. 2023. № 1.
20. Радіолокація — Вікіпедія. *Вікіпедія*. URL: <https://uk.wikipedia.org/wiki/Радіолокація>

21. Конус Маха. www.wiki.uk-ua.nina.az. URL: https://www.wiki.uk-ua.nina.az/Конус_Маха.html
22. АЗК-7 – Вікіпедія. *Bikinedia*. URL: <https://uk.wikipedia.org/wiki/АЗК-7>
23. FDTD (Finite-Difference Time-Domain). *Electromagnetic Template Library* (EMTL). URL: <https://fdtd.kintechlab.com/ru/fdtd>
24. EARS Gunshot Localization Systems. QinetiQ Security & Defence Contractors. URL: <https://www.qinetiq.com/en-us/sectors/defence/land/ears-gunshot-localization-systems>
25. Boomerang State-of-the-Art Shooter Detection. URL: <https://www.raytheonintelligenceandspace.com/capabilities/products/boomerang>
26. Weapon Mounted Acoustic Gunshot Detector | Metravib Defence. Metravib Defence. URL: <https://www.metravib-defence.com/our-solutions/pearl-soldier-protection>
27. Microflown AVISA acoustic vector sensor technology - Microflown AVISA. Microflown AVISA. URL: <https://www.microflown-avisa.com/technology>
28. Ultra Electronics Gunfire Locator – PDFSLIDE.TIPS. *pdfslide.tips*. URL: <https://pdfslide.tips/documents/ultra-electronics-gunfire-locator.html>
29. Київський професійно-педагогічний фаховий коледж імені Антона Макаренка. URL: <https://kppk.com.ua/ELLIB/ebook/Gorbenko/IKT/3/3.htm>
30. Системне програмне забезпечення - Програмне забезпечення. *Google Sites*. URL: <https://sites.google.com/site/programnezabezpecenna/sistemne-programne-zabezpecenna>
31. Difference between System Software and Application Software. *Guru99*. URL: <https://www.guru99.com/difference-system-software-application-software.html>
32. Що таке прогресивні веб-додатки [Електронний ресурс] - Режим доступу до ресурсу: <https://smile-ukraine.com/ua/pwa/introduction>
33. Web-додатки – переваги і недоліки [Електронний ресурс] - Режим доступу до ресурсу: https://mydiv.net/arts/view-web-dodatki_perevagy_i_nedliki.html
34. Розробка Web-додатків [Електронний ресурс] - Режим доступу до ресурсу:

<http://clashdash/directions/web-applications>

35. Що таке Progressive Web App та в чому їх переваги [Електронний ресурс] - Режим доступу до ресурсу: <https://timeweb.com/community/articles/chtotakoeprogressive-web-apps>

36. Тестування ПЗ [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.

37. Майерс Гленфорд Дж. Мистецтво тестування програм: Фінанси та статистика, 1982. — 114 с.

38. Harris Drucker, Yann Le Cun. Improving Generalization Performance Using Back-propagation / Harris Drucker, Yann Le Cun // IEEE Transactions on Neural Networks. - 1992. - Vol.3, №5 - p. 991-997.

39. Build on BNB Chain [Електронний ресурс] - Режим доступу: <https://docs.bnbchain.org/docs/bnbIntro>

40. Getting started with WebStorm [Електронний ресурс] : echo / Режим доступу: <https://www.jetbrains.com/help/webstorm/getting-started-withwebstorm.html>

41. Visual Studio Code Tips and Tricks [Електронний ресурс]. – Режим доступу: до ресурсу: <https://code.visualstudio.com/docs/getstarted/tips-and-tricks>

42. How to Use Sublime Text: A Quick Overview [Електронний ресурс]. – Режим доступу до ресурсу: <https://kinsta.com/blog/how-to-use-sublime-text/>

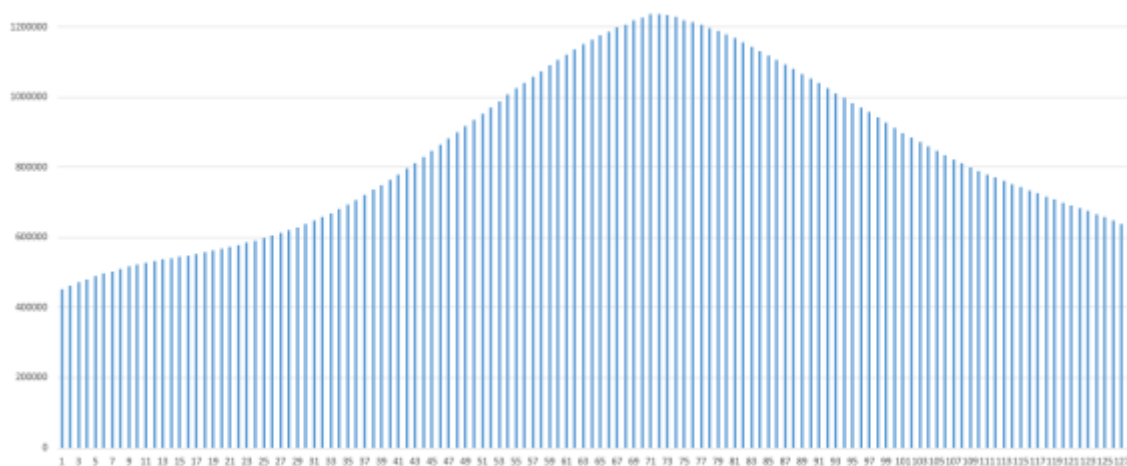
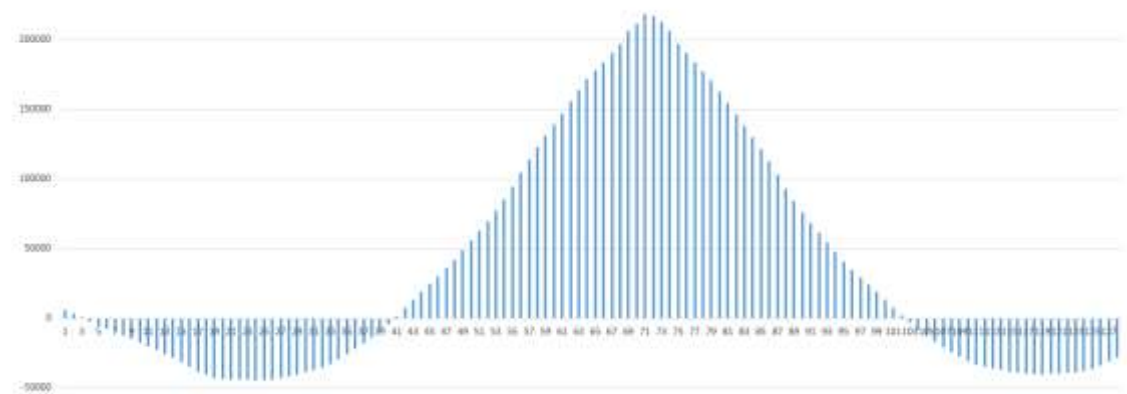
43. Java is a powerful general-purpose programming language. [Електронний ресурс]. – Режим доступу: <https://www.programiz.com/javaprogramming>

44. «CPU vs. GPU Rendering: Which Should You Choose and Why?». IncrediBuild. <https://www.incredibuild.com/blog/cpu-vs-gpu-rendering-whichshould-you-choose-and-why>

45. «CPU vs. GPU Renderer: Which is Better?» AcademyOfAnimatedArt. <https://academyofanimatedart.com/cpu-vs-gpu-renderer-which-isbetter/#:~:text=Modern%20GPUs%20offer%20superior%20processing,times%20faster%20than%20CPU%20rendering>

ДОДАТОК А

Результати розрахунків ВКФ для зміщених сигналів



ДОДАТОК Б

Фрагмент лістингу програмного забезпечення

```

using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.IO;
namespace iTellUtils
{
public class Vertex
{
public List<Vertex> next; public List<double> wnext; public double delta; public
double value; public double
threshold;
public Vertex()
{
next = new List<Vertex>(); wnext = new List<double>();
Random rand = new Random((int)DateTime.Now.Ticks); threshold =
rand.NextDouble() * 0.4;
delta = 0.0;
value = 0.0;
}
}
public class Layer
{
public List<Vertex> vertex; public Layer(int size)
{
vertex = new List<Vertex>(); for (int i = 0; i < size; ++i)
vertex.Add(new Vertex());
}
}
public class NeuralNetworkException: Exception
{
public NeuralNetworkException(string message)
: base(message) { }
}
public class NeuralNetwork
{
private const double speed = 0.05; private List<Layer> layers;
static public NeuralNetwork Current = null; public int getDesision() {
double best = -1e100;
int bestv = -1; double[] x = Output();
for(int i=0; i<x.Length; ++i)

```

```

{
    if (x[i] > best)
    {
        best = x[i]; bestv = i;
    }
}
return bestv;
}
public double[] Output()
{
    double[] result = new double[layers[layers.Count - 1].vertex.Count];
    for(int i=0; i<result.Length; ++i)
    {
        result[i] = layers[layers.Count - 1].vertex[i].value;
    }
    return result;
}
public void Save(string filename)
{
    StreamWriter sw = null; try
    {
        sw = new StreamWriter(filename);
        sw.WriteLine(layers.Count);
        for (int i = 0; i < layers.Count; ++i) sw.Write(layers[i].vertex.Count.ToString() + " ");
        sw.WriteLine();
        for (int i = 0; i < layers.Count; ++i)
        for (int j = 0; j < layers[i].vertex.Count; ++j)
        {
            Vertex v = layers[i].vertex[j];
            sw.Write(v.threshold.ToString() + " " + v.next.Count.ToString() + " "); for (int k = 0;
            k < v.next.Count; ++k)
            sw.Write(v.wnext[k]);
            sw.Write(" ");
        }
        sw.WriteLine();
    }
    catch (Exception e)
    {
        throw new NeuralNetworkException(e.Message);
    }
    finally
    {
        if (sw != null)

```

```

{
sw.Flush();
sw.Close();
}
}
}
public void Load(string filename)
{
StreamReader sw = null; try
{
sw = new StreamReader(filename); string s = sw.ReadToEnd();
string[] _ss = s.Split(' ', '\n', '\r'); List<string> ss = new List<string>(); for (int i = 0; i
< _ss.Length; ++i)
if (_ss[i].Length > 0) ss.Add(_ss[i]);
int p = 0;
int n = int.Parse(ss[p++]); int[] size = new int[n];
for (int i = 0; i < n; ++i) size[i] = int.Parse(ss[p++]);
layers = new List<Layer>(); for (int i = 0; i < n; ++i)
layers.Add(new Layer(size[i]));
for (int i = 0; i < n - 1; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j)
for (int k = 0; k < layers[i + 1].vertex.Count; ++k)
layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);
for (int i = 0; i < n; ++i)
{
for (int j = 0; j < size[i]; ++j)
{
Vertex v = layers[i].vertex[j]; v.threshold = double.Parse(ss[p++]); int m =
int.Parse(ss[p++]);
for (int k = 0; k < m; ++k) v.wnext.Add(double.Parse(ss[p++]));
}
}
}
catch (Exception e)
{
throw new NeuralNetworkException(e.Message);
}
finally
{
if (sw != null) sw.Close();
}
}
public NeuralNetwork(string filename)
{

```

```

Load(filename);
}
public NeuralNetwork(int[] sizes)
{
    Random rand = new Random((int)DateTime.Now.Ticks); layers = new
    List<Layer>();
    for (int i = 0; i < sizes.Length; ++i) layers.Add(new Layer(sizes[i]));
    for (int i = 0; i < sizes.Length - 1; ++i)
    {
        for (int j = 0; j < layers[i].vertex.Count; ++j)
        for (int k = 0; k < layers[i + 1].vertex.Count; ++k)
        {
            layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);
            layers[i].vertex[j].wnext.Add(rand.NextDouble() * 0.4 -
            0.2);
        }
    }
}
public void clear()
{
    for (int i = 0; i < layers.Count; ++i)
    for (int j = 0; j < layers[i].vertex.Count; ++j) layers[i].vertex[j].value = 0.0;
}
private double sigmolda(double x)
{
    return 1.0 / (1.0 + Math.Exp(-x));
}
public int InputLayerSize { get { return layers[0].vertex.Count; } } public void
Run(double[] input)
{
    clear();
    if (input.Length != layers[0].vertex.Count) throw new Exception("Input layer and
    input has different number of
    items");
    for (int i = 0; i < input.Length; ++i)
    {
        layers[0].vertex[i].value = input[i];
        for (int j = 0; j < layers[0].vertex[i].next.Count; ++j) layers[0].vertex[i].next[j].value
        += input[i] *
        layers[0].vertex[i].wnext[j];
    }
    for (int t = 1; t < layers.Count; ++t)
    for (int i = 0; i < layers[t].vertex.Count; ++i)
    {

```

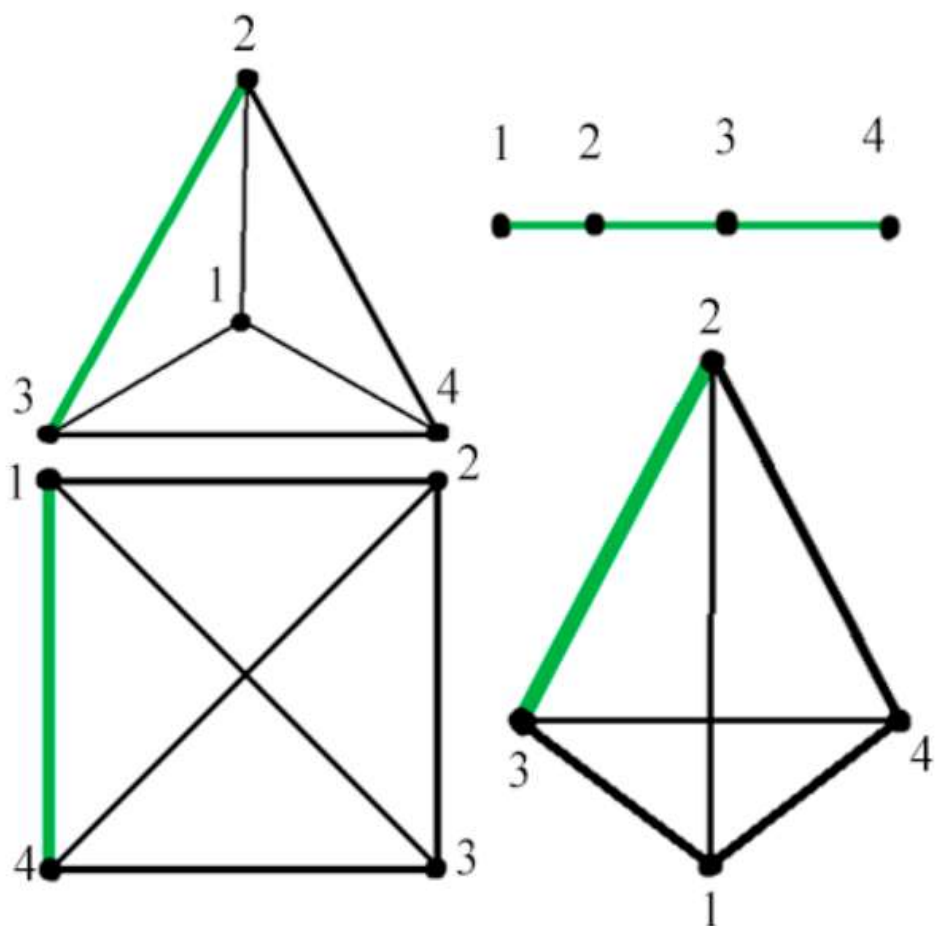
```

Vertex v = layers[t].vertex[i];
v.value = sigmoid(v.value - v.threshold); for (int j = 0; j < v.next.Count; ++j)
v.next[j].value += v.value * v.wnext[j];
}
}
public void Learn(double[] input, double[] need)
{
if (need.Length != layers[layers.Count - 1].vertex.Count)
throw new Exception("Test is something different from expected"); Run(input);
for (int i = 0; i < layers[layers.Count - 1].vertex.Count; ++i)
{
Vertex v = layers[layers.Count - 1].vertex[i];
v.delta = v.value * (1.0 - v.value) * (need[i] - v.value);
}
for (int t = layers.Count - 2; t >= 0; --t) foreach (Vertex v in layers[t].vertex)
{
v.delta = 0.0;
for (int i = 0; i < v.next.Count; ++i) v.delta += v.next[i].delta * v.wnext[i];
v.delta *= v.value * (1.0 - v.value);
}
}

```

ДОДАТОК В

Розроблені конфігурації



ДОДАТОК Г

Вигляд інтерфейсу користувача

