НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Генерація гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI»

Виконав:
студент IV курсу, групи ТР-71
Іванов Валерій Олегович _____

Керівник:
к.т.н, доцент каф. АПЕПС
Лабжинський Володимир Анатолійович _____

Рецензент:
к.т.н, доцент каф. АЕС і ІТФ
Лебедь Наталія Леонідівна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Іванову Валерію Олеговичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи: _Генерація гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI_____

керівник роботи: _Лабжинський Володимир Анатолійович, к.т.н., доцент____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи_ Мова C# у середовищі Visual Studio 2019, .NET Core, Message Passing Interface_____
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести аналіз існуючих програм для генерації гідроакустичного

сигналу, розробити програму для генерації гідроакустичного сигналу з застосуванням технології MPI, налагодити зв`язок комп`ютерів у мережі, виконати програму використовуючи потужності кількох комп`ютерів_____

5. Перелік ілюстративного матеріалу

«Актуальність розробки», «Мета та завдання роботи», «Однорідний океан___ постійної глибини», «Променеве представлення», «Коливальна швидкість»,__ «Розроблений додаток», «Висновки»_____

6. Дата видачі завдання ”___”_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	01.03.2021	
2.	Вивчення та аналіз задачі	03-31.03.2021	
3.	Розробка архітектури та загальної структури системи	01-18.04.2021	
4.	Розробка структур окремих підсистем	19-25.04.2021	
5.	Програмна реалізація системи	26.04-13.05.2021	
6.	Оформлення пояснювальної записки	06.05-01.06.2021	
7.	Захист програмного продукту	12.05.2021	
8.	Передзахист	26.05.2021	
9.	Захист	16.06.2021	

Студент _____ Іванов В.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Лабжинський В.А._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення програмного продукту для генерації гідроакустичного сигналу з застосуванням технології MPI. Об’єктом дослідження є способи та алгоритми моделювання сигналів. Було виконано огляд існуючих програмних застосунків для моделювання сигналів та ознайомитися із проблемами моделювання гідроакустичних сигналів, розроблено програмний продукт генерації гідроакустичних сигналів, який реалізовано методом уявних джерел для розрахунку поля тиску. Створений програмний продукт може бути використаний, як частина системи для моделювання гідроакустичних об’єктів та для наукових досліджень. Загальний обсяг роботи: 68 сторінок, 19 ілюстрацій, 21 бібліографічних посилань та 3 додатки.

Ключові слова: гідроакустичний сигнал, моделювання гідроакустичних сигналів, метод уявних джерел, модуль генерації сигналів, технологія паралельних обчислень.

ABSTRACT

The purpose of the thesis is to create a program product for generating a hydroacoustic signal using MPI technology. The objects of research are the methods and algorithms of signal simulation. An overview of the existing software applications for simulation of signals and the problems of modeling of hydroacoustic signals was performed, the program software of generation hydroacoustic signals, implemented by the imaginary sources for calculating the field of pressure. The created program product can be used as part of the system for modeling hydroacoustic objects and for scientific research. Total volume of work: 68 pages, 19 illustrations, 21 bibliographic references and 3 appendices.

Keywords: hydroacoustic signal, modeling of hydroacoustic signals, method of imaginary sources, module of signal generation, parallel computation technology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГАС — гідроакустична система.

ГІС — геоінформаційна система; поєднує карту території та різні дані, в основному табличного типу.

MPI (Message Passing Interface) — Інтерфейс передачі даних, одна з технологій паралельних обчислень.

CDT (conductivity, depth, and temperature) — це інструмент океанографії, який використовується для вимірювання провідності, глибини і температури водойми.

ВСТУП

У гідроакустичних інформаційних системах передача інформації на відстань, а також одержання інформації про різні об`єкти у водному середовищі здійснюється за допомогою гідроакустичних сигналів, які формують і випромінюють сигнали із заданими характеристиками, змінюють залежно від виконуваних завдань параметри випромінюючих і прийомних антен, а також алгоритми оброблення інформації.

Гідроакустичні сигнали — це змінні в часі й просторі акустичні коливання, що поширюються у водному середовищі й несуть інформацію про виявлюване джерело, а також використовуються для ехолокації об`єктів, навігації й зв`язку.

Генерація гідроакустичного сигналу це складна обчислювальна задача, розв’язання її вимагає виконання великого обсягу обчислень, тому в даному випадку є доцільним застосування технології паралельних обчислень. Багатопроменева структура гідроакустичного сигналу дає можливість знаходити характеристики кожного променя незалежно від інших променів.

Будова паралельних комп`ютерів за існування комп`ютерної промисловості розвилася неймовірними темпами і в самих різних напрямках щоб дозволити виконування поставленої задачі обчислення одночасно на декількох комп’ютерах, а розподілення роботи між ними і дозволяє зробити Message Passing Interface (MPI).

1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Гідроакустика

Гідроакустика — це розділ акустики, який вивчає випромінювання, прийом і поширення звукових хвиль в реальному водному середовищі (в океанах, морях, озерах, річках і т. д.) для цілей підводної локації, зв’язку і т. ін.

Основна особливість підводних звуків це їх слабке загасання, унаслідок чого під водою звук може поширюватись на значно більшу відстань, аніж, наприклад, у повітряному середовищі [1]. Крім загасання, обумовленого властивостями самої води, на дальність поширення звуків під водою впливають рефракція звуку, його розсіяння а також поглинання різними неоднорідностями середовища. Тривалі дослідження процесів поширення звукових хвиль у двовимірних і тривимірних підводних неоднорідних хвилеводах отримали значний імпульс, починаючи з 80х років минулого століття в зв’язку з потребами дистанційного зондування і акустичного моніторингу регіонів Світового океану [2].

Майже у всіх видах сигналізацій, зв’язку, ехолокації, дистанційного дослідження водойм і дна океану використовуються звукові хвилі. При проектуванні інформаційно-вимірювальних систем також застосовуються результати моделювання хвильових процесів та надають змогу вирішити широке певні взаємопов’язані задачі, такі як формування певних структур акустичних полів в заданих хвилеводах, створенні гідроакустичних антен та ін. Тому проблеми створення і вдосконалення математичних моделей акустичних процесів в океані є досить актуальними, також присутня потрібність розробки спеціалізованих програмних продуктів.

1.2. Технології гідроакустики

Звукові хвилі — єдиний вид хвиль, що мають змогу поширюватися у морському середовищі без великого загасання на значні відстані, тому на їхній основі створюється дистанційний метод і технологія вивчення рельєфу дна, дослідження складу і різноманітних властивостей морського ґрунтів і порід, визначення глибини і виявлення рухомих і нерухомих об’єктів, пошук місцезнаходжень затонулих об’єктів за заданими характеристиками, перевірка стану підводних технічних споруд, дослідження стану підводних частин технічних споруд, дослідження середовища та інше [3].

Переважна більшість гідроакустичних систем має дуже схожі складові та використовуються у вирішенні багатьох задач, що призводить до зацікавленості в взаємодії між різними корпораціями, які так або інакше, працюють в напрямах застосування та дослідження гідроакустичних технологій [4].

Гідроакустичні засоби (датчик) створюються для виконання певних цілей, наприклад:

— геоакустичних;

— знаходження ресурсів(сировинних і біологічних);

— моніторинг морського середовища;

— моніторинг стану підводних трубопроводів;

— боротьба з підводними мінами;

— розпізнання підводних човнів.

Методи гідроакустики розробляються та реалізуються за допомогою технічних пристроїв та систем. Гідроакустичні пристрої це сукупність елементів, принцип роботи яких базується на використанні властивостей звукових хвиль у водяному середовищі. Враховуючи особливості водяного середовища для моніторингу використовують різні методи [5].

Гідроакустика передбачає генерацію сигналу, який показує зміну акустичного тиску води, що створюються звуковою хвилею. У воді існує шар де звуковий рух є більш повільним, проте дуже ефективним, даний шар називають звукоізоляційним підводним каналом, який як правило знаходиться на глибині приблизно 1000 м, та утворює своєрідний хвилевід. Вище та нижче від хвилеводу швидкість звуку змінюється через вплив температур, а усередині — сигнали відбиваються від країв каналу назад у його межі і таким чином поширюються на дуже велику відстань. Також гідроакустика є однією з чотирьох технологій, що використовуються міжнародною системою моніторингу якості дотримання договору про всесвітню заборону ядерних випробувань. Ядерні вибухи під водою, у атмосфері, коло поверхні води або під землею біля берегу генерують звукові хвилі, які можуть бути виявлені на відстані в сотні тисяч кілометрів. Як наслідок, лише одинадцять станцій на весь світ є достатніми для моніторингу великих океанів Землі, з акцентом у південній півкулі, де в основному домінує вода.

Для прикладу розглянемо роботу гідролокатора (рис. 1.1), що заснований на вимірюванні часу, за який акустичний сигнал проходить від випромінювача до дна, а його ехо-сигнал, відбивається з певним імпульсом, коли зустрічає на своєму шляху об’єкт. При відомому часі проходження імпульсу від випромінювача до об’єкту, часі повернення ехо-сигналу від об’єкта до приймача та швидкості поширення звуку у воді можна визначити відстань між ними. Метод визначення відстані між об’єктами у воді по часу проходження акустичного імпульсу застосовується в різноманітних акустичних приладах.

Залежно від цілей гідролокатори бувають різні: для дистанційного дослідження складу верхнього шару ґрунту, для сейсмічного зондування, для огляду дна, навігація для визначення координат, для знаходження рибних скупчень, що встановлюються на риболовних суднах та багато іншого.

Очевидно, що гідроакустика знайшла широке застосування, та має високий пріоритет в наукових дослідженнях.

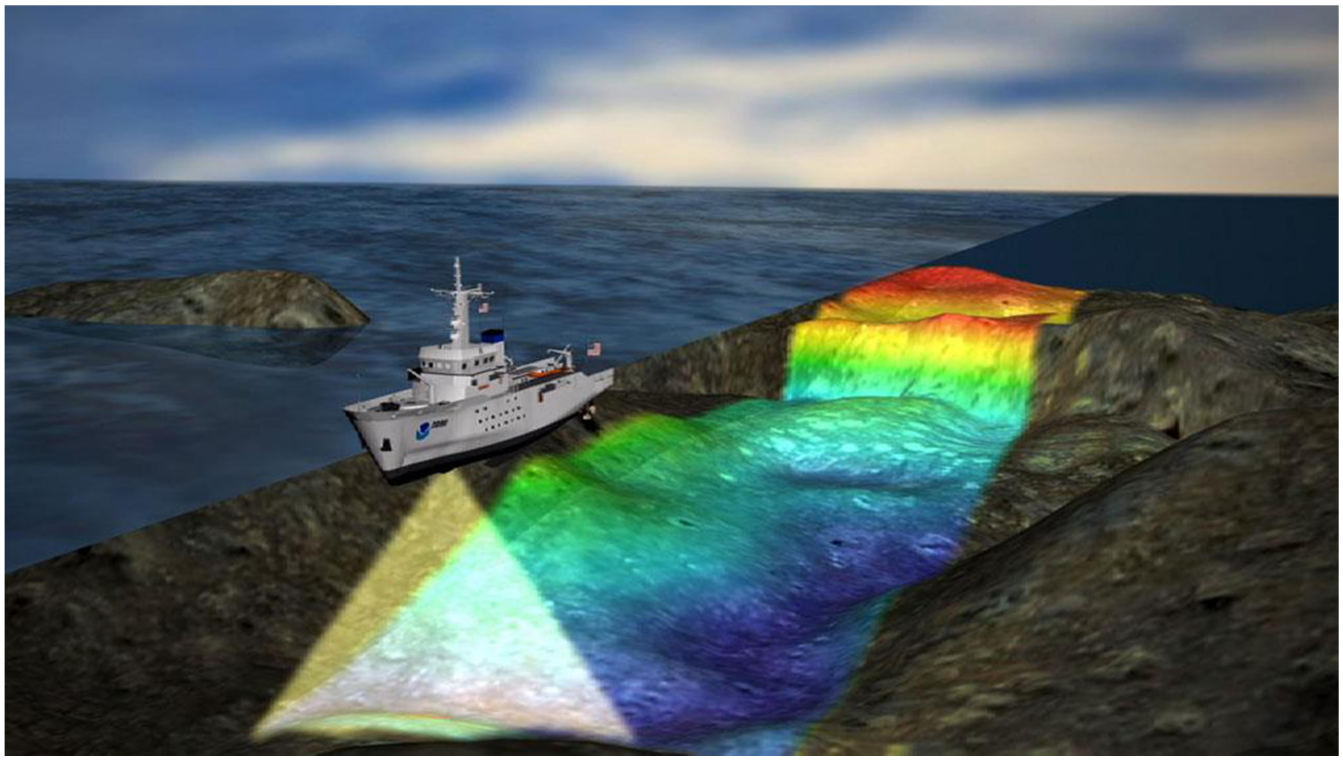

Рисунок 1.1 — Принцип роботи гідролокатора

1.3. Програмні продукти для моделювання сигналів Actran та Adams

У теперішній час, найбільш розповсюдженим програмним продуктом який використовується, у переважній більшості випадків для моделювання сигналів є Actran (рис. 1.2).

Розробка Actran почалася, у наш час, професором Луврської інженерної школи Жан-Луї Міхеотом та професором в університеті Брюсселя і колишнім президентом Королівської академії наук Жан-П'єром Котом. Початкова ідея була розробкою інструментального засобу моделювання джерел акустичного сигналу на основі наявних елементів для віброакустичних застосувань, які б

змогли подолати обмеження, на той час, домінуючого методу скінченних елементів [13].

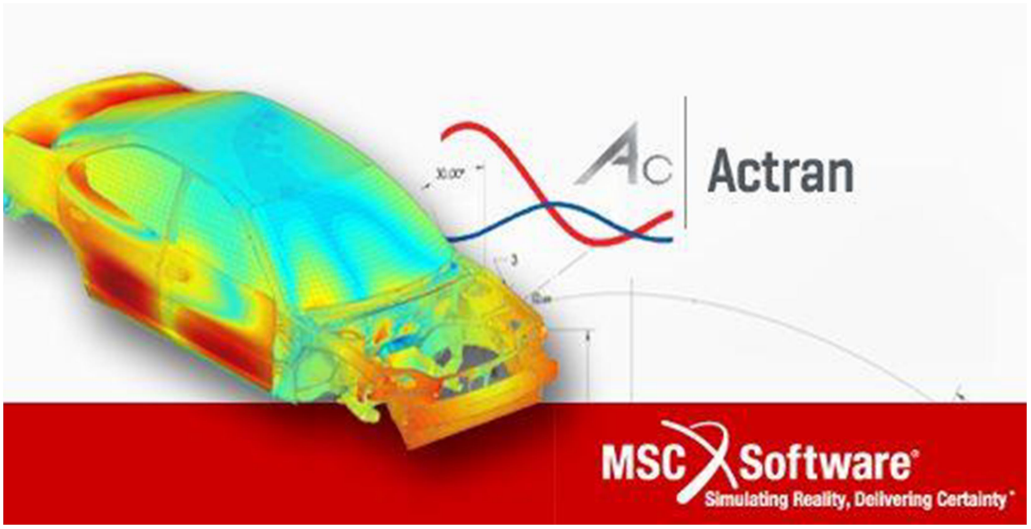

Рисунок 1.2 — Логотип Actran

Використання нескінченних елементів дозволяло моделювати складні джерела звуку, поєднання декількох матеріалів в одній моделі і оброблення моделі з декількома мільйонами ступенів свободи. Початковою цільовою програмою було прогнозування звукової передачі через складні перешкоди.

Головною особливістю Actran є використання нескінченних елементів, як альтернатива скінченним елементам для моделювання невідповідних граничних умов і обчислення акустичного поля. Перший комерційний випуск був широко розповсюджений у 2002 році. Actran була розроблена мовами Python і C ++ і є сумісним з операційними системами Linux та Windows.

В основі Actran лежать методи обох - скінченних і нескінченних елементів, що дозволяють моделювати звукові ефекти як у закритих територіях, так і в відкритому просторі, враховуючи умову невідбиття звуку на межах

області, що розраховується. Багата бібліотека елементів і матеріалів надає широкі можливості моделювання. Бібліотеку елементів складають одновимірні, двовимірні та тривимірні елементи для розв'язування тривимірних, двовимірних і осе-симетричних задач, багатоточкових з'єднань, нескінченних елементів.

Бібліотека наявних матеріалів дає змогу моделювання звукових середовищ з урахуванням загасання і втрат у тонких шарах. Як зовнішні навантаження, та граничні умови, розглядаються джерела звуку різних типів, моди коливань звукового середовища в каналах постійного перерізу, граничні умови по тиску, швидкості, прискоренню.

Adams — комплексне програмне рішення для створення віртуальних прототипів складних динамічних систем (рис. 1.3)

Рисунок 1.3 — Логотип Adams

Adams — повний комплект видів кінематичних зв’язків з лінійними і не лінійними характеристиками, комплект навантажень, комплект кінематичних впливів, доступних користувачу для побудови розрахункової моделі що відповідає реальності;

— простота вивчення і використання, оскільки дослідження віртуального прототипу в Adams відповідає основним етапам роботи із зразками

— інтуїтивно зрозумілий інтерфейс - якщо користувач знайомий з іншими програмними засобами CAE або CAD, то дуже швидко опанує роботу і з Adams;

— ефективні засоби відображення результатів моделювання, присутні анімація і побудова графіків;

— можливість зміни параметрів розрахункової моделі — зміна параметрів призводить до зміни характеристик моделі та її зміни, параметри моделі можуть бути зв'язані функціональними відповідностями тощо;

— наявність надбудов для Adams, наприклад Adams/Car, Adams/TrackedVehicle, Adams/Machinery, дозволяє розширювати спектр вирішуваних завдань, спрощувати взаємодію інженера з розрахунковою моделлю, як на етапі її створення і модифікації, так і на етапі проведення фінальних досліджень; через комплексний підхід при моделюванні виробів у Adams, із використанням гнучких деталей і компонентів, результати обрахування динаміки можна використовувати як історію навантаження і навантажуваних циклів (DutyCycle) для подальшого дослідження;

— додатковий інтерфейс для взаємодії із зовнішніми розрахунковими наборами й проведення спільних розрахунків відкриває можливості з моделювання складних механічних моделей із високонелінійними пружними деталями, підготовленими у програмному комплексі Marc;

— сумісність із програмним забезпеченням для моделювання систем автоматичної регуляції та керування Easy5, Simulink, Functional Mock-up Interface (FMU), із одними програмами дає можливість моделювання і дослідження дуже складних динамічних систем і пристроїв;

— додаткові плагіни дають користувачу можливість прямо взаємодіяти (з інтерфейсу Adams) з КЕ системою MSCNastranEmbeddedFatigue для оцінки вузлів змодельованого виробу (розширення Adams2NEF), з системою Actran для оцінки звукових властивостей (плагін Adams2Actran).

1.4. Постановка задачі

Сигнал у гідроакустичних інформаційних системах є значенням акустичного поля, тобто хвилі, які змінюються у часі та просторі і зображають сигнал сукупністю своїх параметрів.

В загальному випадку кажуть про просторово-часові сигнали, природним носієм яких є векторне акустичне поле, бо переміщення частинок середи описується вектором швидкості коливань. Тому для цього необхідно обчислити векторне поле коливальних швидкостей і скалярне поле тисків. Важливим а також складним в гідроакустиці є складення математичних відношень, що пов'язують характеристики об'єктів з корисними сигналами, тобто розроблення відповідних моделей сигналів, вид яких визначається фізичною суттю процесу, рівнем його пізнання і метою дослідження.

Опис сигнальних полів є функція часових та просторових координат. Після просторового оброблення на приймальні антени, або якщо дана антена має невеликі розміри в порівнянні з довжиною хвилі отриманого коливання в постійному полі тисків, сигнал буде лише функцією від часу.

Вдосконалення математичних, алгоритмічних та програмних гідроакустичних та гідролокаційних систем неможливо відтворити без моделювання роботи їхніх частин на попередніх тестуваннях. Повне тестове покриття в даному випадку багато в чому залежить від достовірності побудованої математичної моделі у гідроакустичних системах із врахуванням обчислювальної продуктивності системи. Саме обмеження обчислювальної потужності комплексів моделювання дуже часто не дає можливості прямого використання математичної моделі у режимі реального часу, тому виникає потрібність розробки паралельних програм для впровадження цих обчислювань на декількох машинах водночас.

2. МОДЕЛЮВАННЯ ГІДРОАКУСТИЧНОГО СИГНАЛУ

2.1. Математичні моделі хвильових процесів

З математичної точки зору поширення гармонійних звукових хвиль у просторі описується крайовими задачами для рівняння Гельмгольца з комплексним несамоспряженим оператором.

У теперішній час для комп'ютерного моделювання звукових полів в підводних неоднорідних хвилеводах використовуються 3 класи математичних моделей у вигляді крайових (початково-крайових) завдань для еліптичних (або параболічних типу Шредінгера) хвильових рівнянь [6].

— крайова задача для еліптичного хвильового рівняння Гельмгольца;

— задачі Коші для еліптичного перетвореного хвильового рівняння Гельмгольца;

— крайові задачі для параболічних хвильових рівнянь Шредінгера з комплексним несамоспряженим оператором.

Для неоднорідного середовища звукове поле точкового гармонійного джерела (функція від часу приймається у вигляді e-iwt) у прямокутній системі координат (x, y, z) описується рівнянням Гельмгольца:

$$p(x)div(p(x)grad(p))+k(n(x)+iv(x))p=-\sigma(x-x_0) \qquad (2.1)$$

де $p(x)$,$x=(x,y,z)$ — акустичний тиск;

$k_0=\omega/c_0$ — хвильове число;

$n(x)=c(x)$,$x=x(x,y,z)$ — коефіцієнт заломлення;

$c(x)$ — швидкість звуку (c_0 — швидкість звуку в деякій точці);

$p(x)$ — густина середовища;

$v(x)\geq 0$ — коефіцієнт затухання;

$x_0=(x_0,y_0,z_0)$ — координати джерела;

$\delta(x-x_0)$ — дельта-функція Дірака;

$i=\sqrt{-1}$ — уявна одиниця [8].

У циліндричній системі координат (r,φ,z) рівняння (2.1) набуває вигляду:

$$pdpdr(rpdpdr)+rpdd\varphi(pdpd\varphi)+pddz(pd\varphi dz)+kn(r,\varphi,z)+iv(r,\varphi,z)p=-\sigma(r)2\pi r\sigma(z-z_0) \quad (2.2)$$

Під час моделювання хвильової задачі широко використовують математичні моделі м'яких, жорстких і імпедансних меж хвилеводів. Імпедансні умови мають комплексні коефіцієнти і описують втрати акустичної енергії. У неоднорідному за густиною середовищі на межі розділу середовищ виконується умова безперервності акустичного тиску і потоку. Крім того, для коректної постановки крайової задачі в необмеженій області у нескінченності необхідно задати умови випромінення, що забезпечують єдність розв'язку. У загальному випадку простору R_n для вирішення рівняння Гельмгольца у середовищах без затухання $(v(x)=0,n(x)=1)$ мають виконуватися умови випромінювання Зоммерфельда:

$$p(x)=O(r_1-n_2),dpdr\pm ikOp=O(r_1-n_2),x=(x,y,z),r=x\to\infty \quad (2.3)$$

де знаки $\mp$ відповідають функції з часом $e\mp i\omega t$.

2.2. Перехід до двовимірних координат

Щоб змоделювати акустичний сигнал, потрібно перейти у двовимірну систему координат. Даний перехід буде залежати від координат джерела звуку та ГАС, яка його приймає.

Наприклад, якщо джерело сигналу завжди лежить в координатах (0, 0, z_0), а система, яка приймає сигнал в координатах (x,y,z), тоді перехід до двовимірних координат буде такий, як на рисунку 2.1.

У цьому випадку нова система координат матиме осі (r,z), де значення z залишається таким же як і в тривимірних координатах, а $r(x,y)$ буде залежати від координати об'єкта по x і y. Модельна система координат розташована в площині між точками розташування джерела, гідроакустичною системою приймача та буде паралельною до осі z.

Джерело матиме координати: $\{\sqrt{02}+ 0, 2=0z0= z0$

Координати ГАС: $\{ r= \sqrt{x2}+ y, 2z= z$

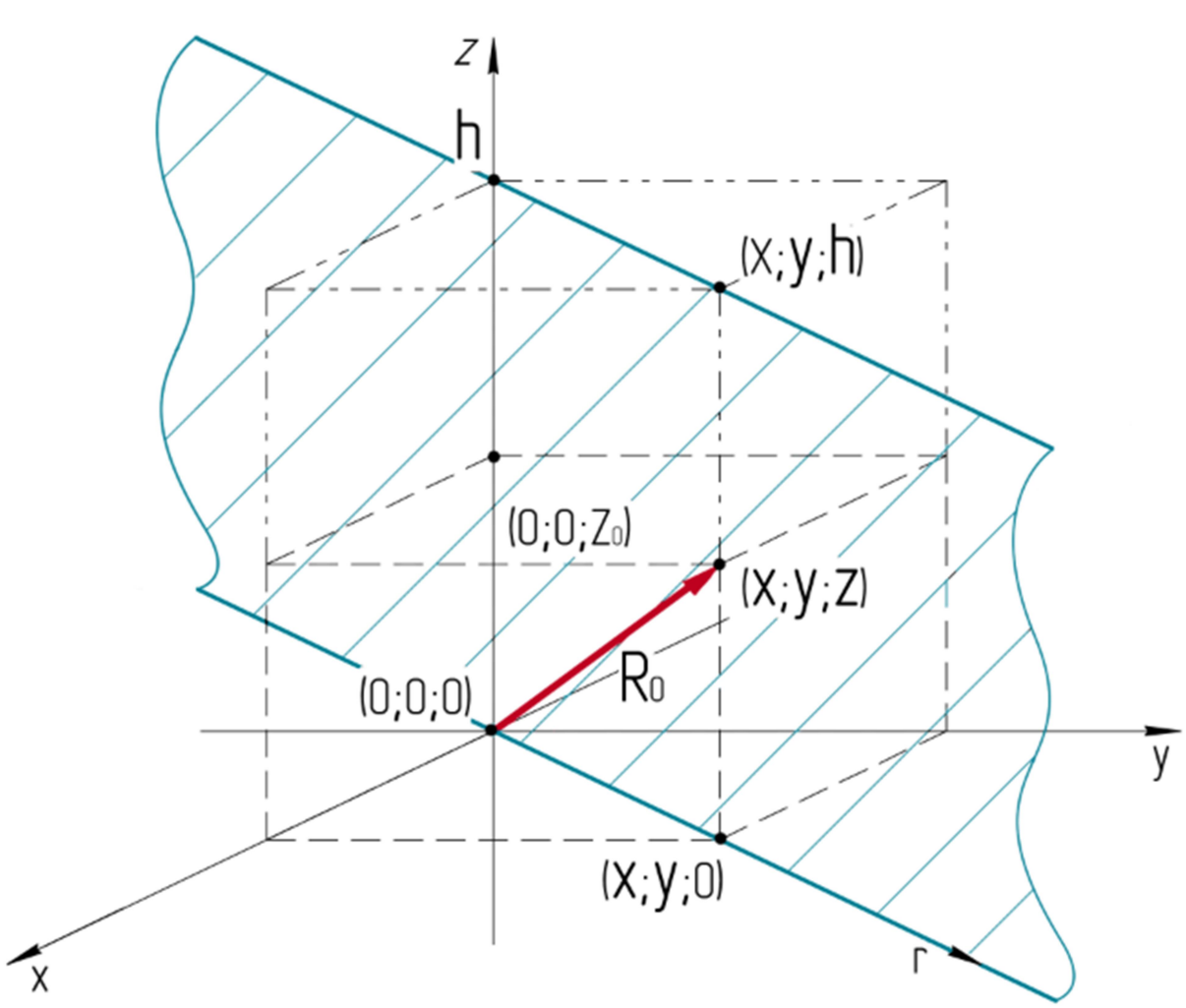

Рисунок 2.1 — Перехід у двовимірну систему координат

2.3. Однорідний океан постійної глибини

Простий профіль швидкості акустичного сигналу має вигляд $c(z)= c_0$, де c_0 — const. В цьому випадку для точкового джерела крайове рівняння прийде до вигляду

$$\Delta p+k2p=-\delta(r)2\pi r\delta(z-z0). \qquad (2.4)$$

Тут координати джерела x_0 це $r=0,z=z_0$. Граничною умовою на поверхні є рівність:

$$p=0 \text{ при } z=0\ . \qquad (2.5)$$

Якщо глибина постійна ($h=const$), граничною умовою дна буде:

$$p_z=0 \text{ при } z=-h\ . \qquad (2.6)$$

Крайові задачі (2.4) - (2.6) із правильною умовою випромінювання визначають поле тиску точкового джерела у однорідному океані сталої глибини із вільною поверхнею та твердим дном. Оскільки задача циліндрично-симетрична, отримане рішення $p(r,z)$ не буде залежати від кутових координат [9].

2.4. Променеве представлення

Достатньо наочним прикладом для $p(r,z)$ є променеве представлення. Для цього випадку розглянемо рівняння (2.4) у тривимірному просторі, ігноруючи

граничні умови (2.5) і (2.6). Загальне сферично-симетричне рішення рівняння (3.4) матиме вигляд

$$p0\ (R)= AeikR/R0+ Be{-}ikR/R0\ ,$$

де $A+B=1/4\pi$,

а $R0=[r2+\ (z{-}z0)2]1/2$ — відстань від джерела.

Щоб вилучити хвилю e_{-ikR}, що приходить до приймача із нескінченності, накладаємо умову випромінювання: $\lim_{R\to\infty}R0[p0'(R){-}\ ikp0(R0)]=0$ (2.7) З цього маємо, що $B=0$ і сферично симетричне рішення (2.8) прийме вигляд $p0(R0)= e_{ikR}/4\pi R0$ (2.9). Експоненту в (2.9) можна інтерпретувати як добуток ik і фазової функції R. Ця фаза дорівнює нулеві у джерелі і зростає пропорційно відстані вздовж прямих ліній, які поєднують джерело із точкою поля. Ми називаємо цю пряму лінію променем. Множник $1/R0$ це амплітуда. Він спадає протилежно пропорційно квадратному кореню від площі поперечного перерізу променевої трубки, бо ця площа зростає як $R02$. Коли деякі промені падають на верхню границю $z=0$, тоді виникають відбиті промені.

Амплітуду відбитої звукової хвилі треба помножити на коефіцієнт відбиття, що рівний -1, для того щоб сума падаючої й відбитої хвиль відповідала умові (2.5). Так як усі промені виглядають так неначе прийшли від уявного джерела, що знаходиться у точці $r=0$, $z=-z0$, то задана побудова дає відбиту хвилю вигляду $-e_{ikR'}/R'$, де R' це і є відстань від уявного джерела.

Така ж сама побудова відбувається і для променів, що відбиті від дна водойми. Однак тут коефіцієнт відбиття рівний +1, бо сума падаючої і відбитої хвиль мають задовольняти умову (2.6). Уявне джерело сигналу буде розміщене у точці $z= -z0-2h$. Саме тому відбита від дна хвиля матиме вигляд $e_{ikR''}/R''$, де R'' це відстань від уявного джерела.

Багаторазове відбиття від стінок хвилеводу породжує групи променів, кожна з яких виглядає, неначе прийшла з якоїсь уявної точки $z= \mp z_0+ 2nh$, $n=0,\mp1,\dots$. Враховуючи кількість відбить, отримуємо вираз загального поля p у вигляді суми падаючої хвилі та відбитих хвиль:

$$p(r,z)=\frac{1}{4\pi}=\sum(-1)^n\left\{\frac{e^{ik[r^2+(z-z_0-2nh)^2]^{\frac{1}{2}}}}{[r^2+(z-z_0-2nh)^2]^{\frac{1}{2}}}-\frac{e^{ik[r^2+(z+z_0-2nh)^2]^{\frac{1}{2}}}}{[r^2+(z-z_0-2nh)^2]^{\frac{1}{2}}}\right\} \qquad (2.10)$$

Ця формула і є променевим представленням рішення p(2.10), що відповідає багатократному відбиванню, бо в ньому кожний доданок, окрім члена $n=0$, описує деяку хвилю, що відбилась деяку кількість разів від поверхні та дна хвилеводу.

На рисунку 2.2 зображено деяке точкове джерело, що заходиться у точці z = z0, і випускає промені у всіх напрямках.

Чотири промені потрапляють у точку (r,z).

Перший промінь із довжиною R є прямим.

Другий промінь із довжиною R' відбився від границі поверхні $z = 0$ та має вигляд, неначе він прийшов із джерела, що знаходиться у точці $z = -z_0$.

Третій промінь із довжиною R'' вже відбився від дна і має вигляд, неначе прийшов із джерела в точці $z = -2h-z_0$.

Четвертий промінь відбитий спочатку від поверхні хвилеводу, потім від дна водойми і виглядає як промінь, який прийшов із джерела у точці $z = -2h+z_0$. Таким чином розраховують довжину n—го променя.

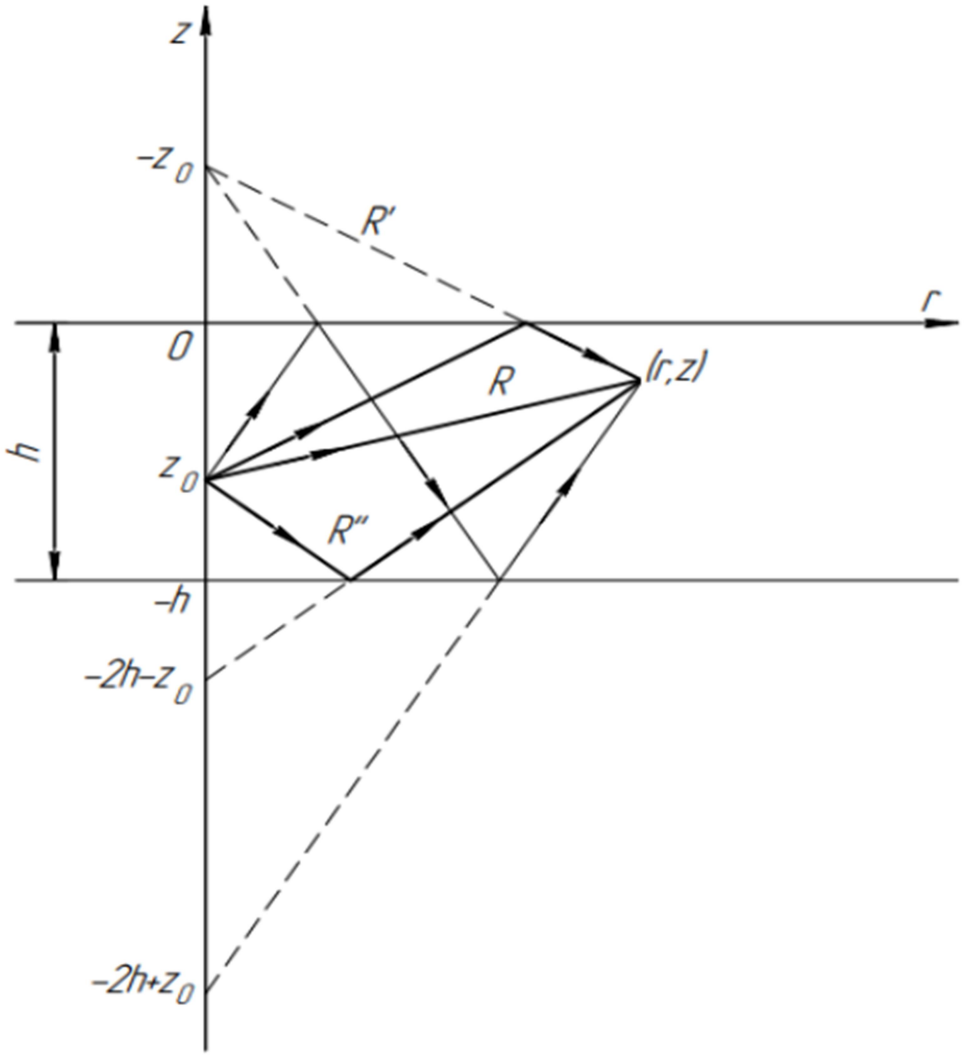

Рисунок 2.2 — Координати уявних джерел

2.5. Розрахунок коливальної швидкості

Для генерації сигналу, окрім сумарного звукового тиску, розраховується іще один фізичний параметр акустичного поля — коливальна швидкість. При розповсюдженні акустичної хвилі, частини повітря починають коливатися відносно положення рівноваги.

Швидкість із якою коливаються частини середи відносно свого положення рівноваги, називається коливальною. Коливальна швидкість це похідна величина від звукового тиску, якщо звуковий тиск обраховується за допомогою функції косинус, то коливальна швидкість є функцією синус та рахується за формулою:

$$\mathrm{V}x_{,,} = sin\ (\omega t + \varphi(R)\ |Vx,y,z|\ R0 \qquad (2.11)$$

де ω — кутова частота, t — час, $|Vx,y,z|R0$ — відношення довжини проекції коливальної швидкості на відповідну вісь до відстані від джерела до ГАС

Гармонічне коливання може бути отримане із обертання вектора коливальної швидкості по колу. Кут повороту коливального вектору для n-го променя у момент потрапляння до приймача це його фаза.

Наприклад, протягом одного періоду фаза змінюється від 0 до 2π. Це відповідає тому що вектор зробить повний оберт проти часової стрілки, а для n-го променю “надлишкова” фаза розрахується, як залишок від ділення довжини даного променя на довжину хвилі.

Для знаходження коливальної швидкості, розраховується значення фаз для кожного променя, та її сумарне значення (рис. 2.3).

Розрахунок модулю коливальної швидкості є векторним та розраховується по правилу паралелограма, де довжина кожного вектору нормується за амплітудою, а кут прийме значення фази (рис. 2.4).

Після обрахунку коливальної швидкості та сумарного тиску, отримані дані характеризуватимуть гідроакустичне поле сигналу в межах акваторії за заданими параметрами.

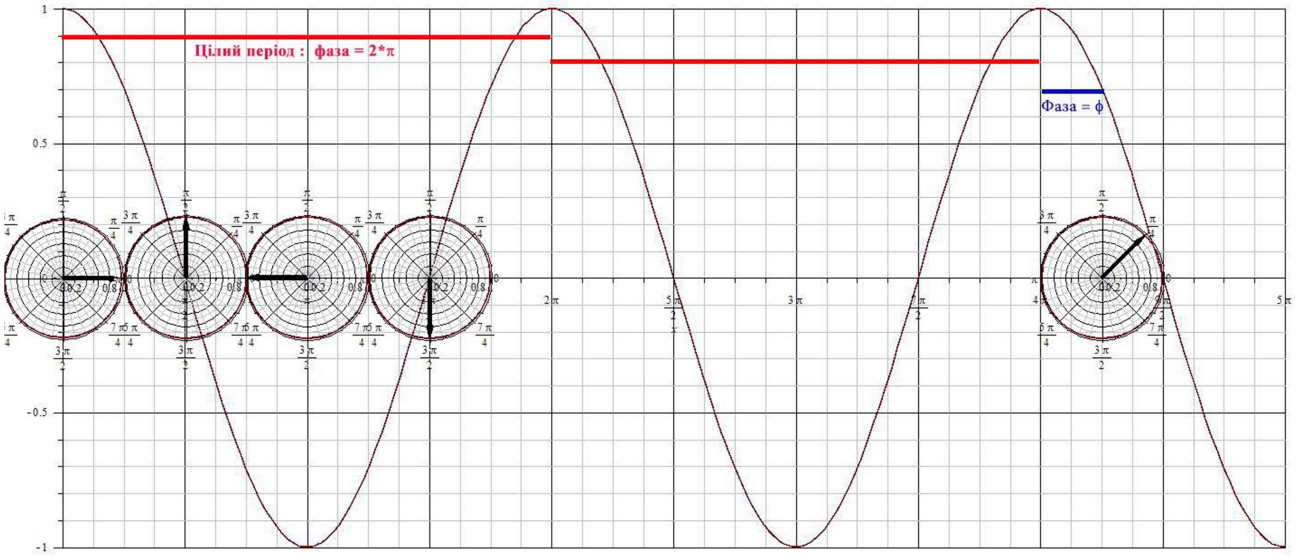

Рисунок 2.3 — Пояснення до обрахунку фази

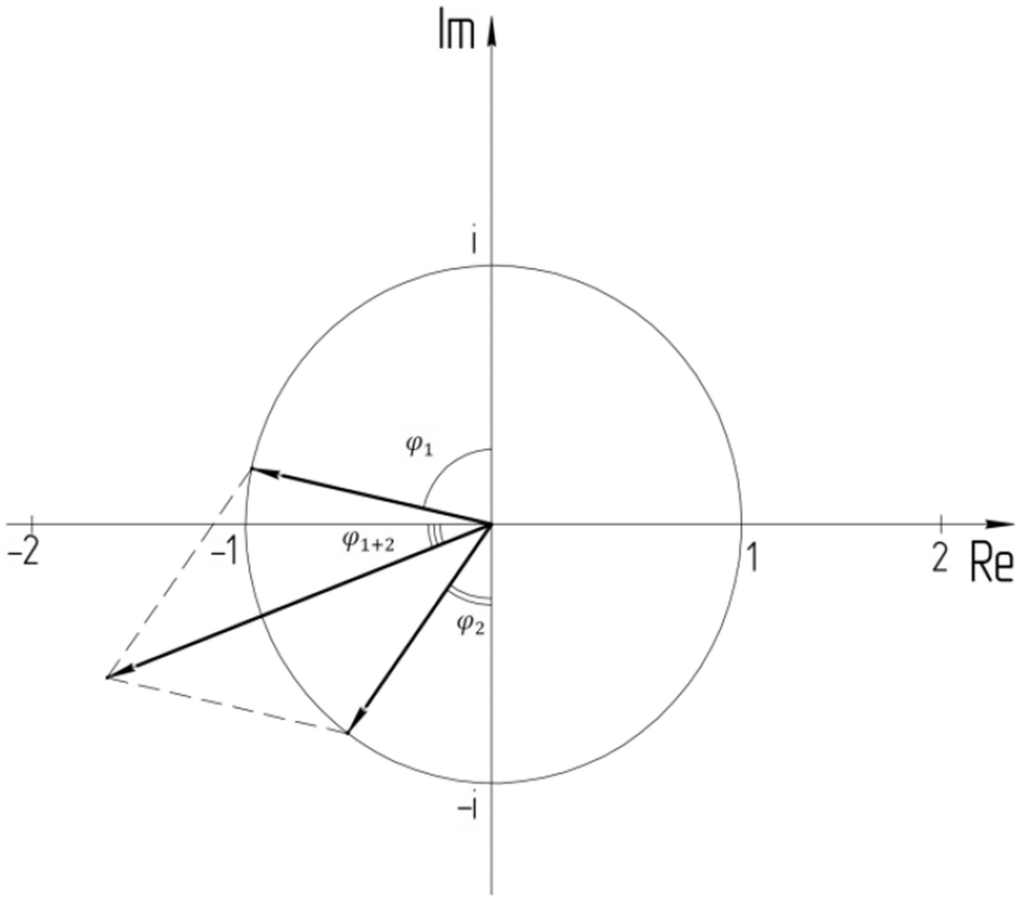

Рисунок 2.4 — Розрахунок модуля коливальної швидкості

2.6. Альтернативний метод моделювання сигналу

Перед початком моделювання розповсюдження акустичних хвиль треба визначити хвилевід, поділений двома горизонтальними і двома вертикальними положеннями. Зазвичай вертикальне положення вважається поверхнею і дном водойми. Положення джерела акустичного сигналу встановлюється в даному діапазоні.

Із точки зору променевої теорії існує задача моделювання поширення акустичних хвиль. Загальна стратегія трасування променів полягає у наступному:

— установити кут запуску в положенні джерела, і далі простежити промінь для кожного кута запуска, допоки він не буде поглинутий повністю або не вийде з діапазону;

— на кожному кроку визначається координати променів, перевіряється, чи є промінь поза поверхнею чи дном.

Щоб простежити за гідроакустичними променями в деякому діапазоні згідно теорії променів, потрібно розв'язати систему рівнянь ейконала, яка регулює поширення акустичних хвиль.

Проблеми поширення акустичних променів часто включають пунктуальні джерела, що мають циліндричну симетрію, або, щонайменше, вважаються середовищами, які повільно змінюються за азимутом і азимутні залежності можна ігнорувати при обчисленні передачі у вказаному напрямі.

Щоб застосувати цю стратегію необхідно знати профіль швидкості звуку в розглянутому діапазоні і далі визначити координати променів.

Профіль швидкості звуку можна визначити за допомогою прямого вимірювання або за допомогою метода розрахунків інформації з датчика CTD - провідності, температур, глибин.

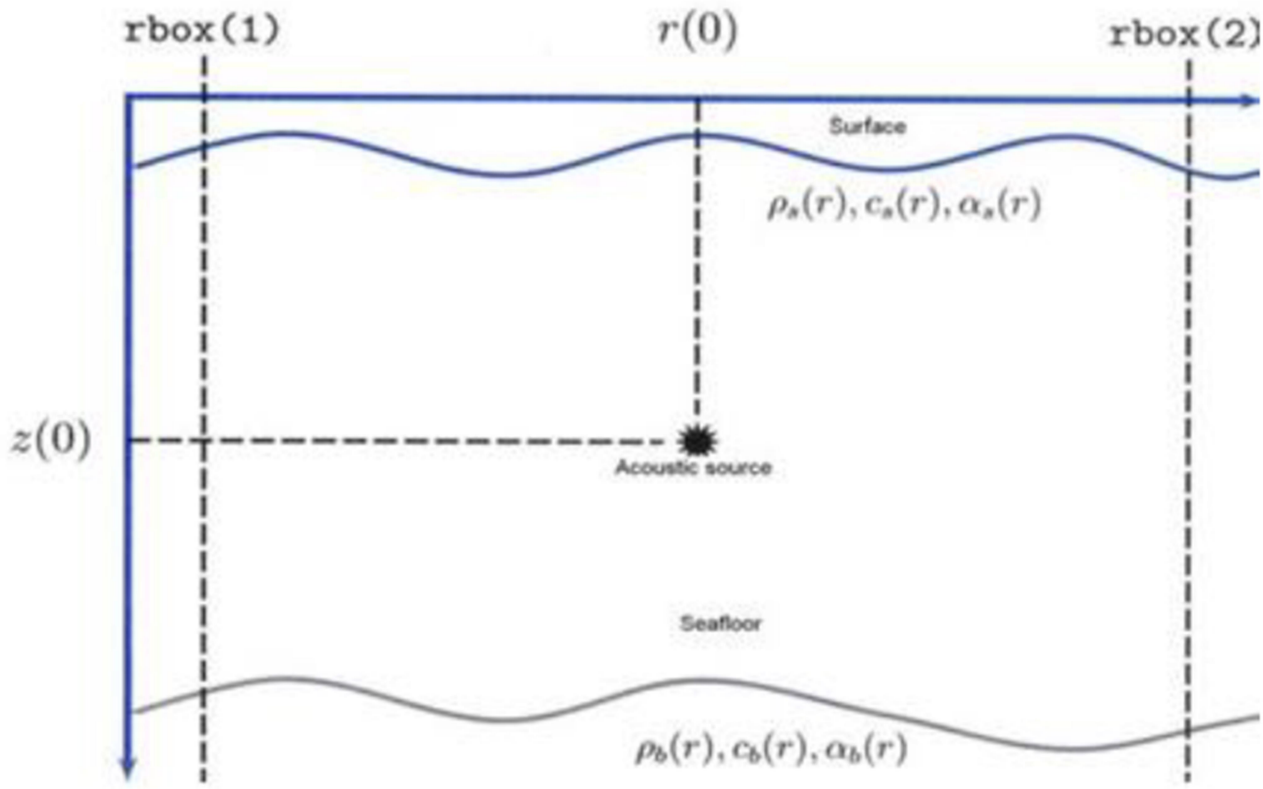

Рисунок 2.5 — Трасування променів водного діапазону

Для того щоб відстежити гідроакустичні промені у деякому діапазоні згідно з теорії променів, треба розв'язати систему рівнянь ейконала, яка регулює поширення акустичних хвиль [12].

Для відстеження гідроакустичних променів у деякому акваторіальному діапазоні треба встановити ряд параметрів.

Параметри ділять на групи:

• горизонтальний діапазон;

• вертикальний діапазон;

• плоска поверхня моря;

• хвиляста поверхня моря;

• типи судових поверхонь;

• горизонтальна координата джерела;

• вертикальна координата;

• частота джерела сигналу;

• кількість відстежених променів;

• перший кут запуску;

• останній кут запуску;

• параметри, що визначають швидкість звуку.

3. ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

При розробці програмних продуктів важливим чинником є вірний вибір технології та засобів програмної реалізації. Середою для створення програмного продукту було Microsoft Visual Studio 2019 (рис 3.1). Для створення інтерфейсу використовувався фреймворк .NET Core. Алгоритми розроблені мовою C# на базі платформи .NET Core. Для паралельних обчислювань було використано MPI.

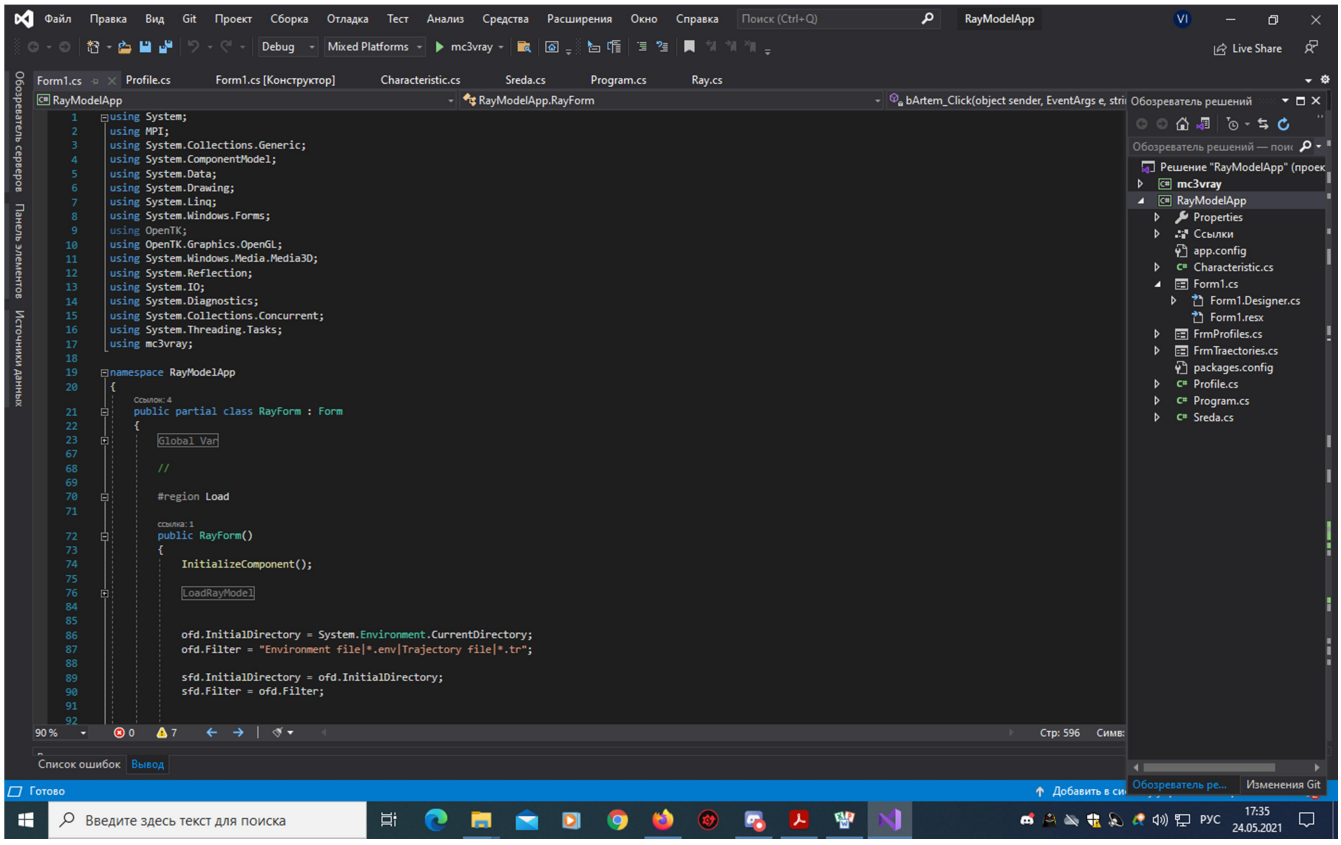

Рисунок 3.1 — Інтерфейс середовища Visual Studio

3.1. Середовище Visual Studio 2019

Visual Studio дозволяє створювати і підключати сторонні застосунки (плагіни) для розширення функціональності практично на кожному рівні,

додавання нових наборів інструментів (для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки) [10].

Visual Studio форматує код у міру його введення, автоматично вставляє потрібні відступи і застосовує кольорове кодування для виділення елементів. Такі незначні поправки роблять код зручнішим для читання і менш схильним до створення у ньому помилок.

Присутні у Visual Studio параметри форматування можна змінювати, що дуже зручно у тих випадках, коли користувач бажає змінити форматування на більш зручне для нього.

Середа Visual Studio спрямована на те, щоб допомагати користувачу у написанні його програми. Присутні функції, що запобігають створенню помилок, напри IntelliSense (пропонує вірні варіанти замість помилкових), функції пошуку і виправлення (дозволяють знайти певне слово у проекті і замінити його) і функції встановлення і видалення коментарів (б приховати блоки коду, коли з ними працювати не треба), дозволяють користувачу працювати, не витрачаючи часу на дрібниці.

Запропоновані в Visual Studio інструменти налаштування є кращим засобом для відстеження невідомих помилок і відслідковування невірної поведінки програми. Користувач може виконувати свій код не одразу а по рядках, встановлюючи точки зупину, при бажанні зберігати їх для використання в майбутньому, і за необхідності переглядати поточну інформацію з пам'яті.

Visual Studio 97 - перша випущена версія Visual Studio, в якій уперше були зібрані разом різні засоби розробки ПЗ. Вона була випущена в двох версіях - Professional і Enterprise, і включала Visual Basic 5.0, Visual C++ 5.0, Visual J++ 1.1, Visual FoxPro 5.0 і середовище розробки ASP, що уперше з'явилося, - Visual InterDev. Visual Studio 97 була першою спробою Microsoft створити єдине середовище для розробки на різних мовах програмування : Visual C++, Visual

J++, Visual InterDev і MSDN використовували одне середовище - Developer Studio. Visual Basic і Visual FoxPro використали окремі середовища для розробки.

3.2. Платформа .NET Core

У теперішній час .NET Core (рис 3.2) — це одна з найкращих платформ розробки програмного забезпечення із відкритим кодом, яку підтримує Microsoft та спільнота .NET на сайті GitHub. .NET Core це комплексна реалізація бібліотек і серед виконання, в яку включається набір .NET Framework. Вона є багатоплатформеною (підтримує і Windows, macOS і Linux) також використовується для створення програмного забезпечення для приладів, хмар та Інтернету речей.

Платформа розробки .NET Core надає:

— типізацію;

— завантаження бібліотек;

— створення власних бібліотек;

— необхідні базові служби.

Головною особливістю .NET Core є гнучке налаштування — користувач може установлювати платформу чи як частину свого застосунку, чи окремо від нього. Окрім цього, потреби у дисковому просторі і дисковому часі зменшуються, і користувач може запустити додаток .NET Core на будь якій ОС. Повне розгортання має в наявності усі необхідні компоненти і бібліотеки.

Цей тип розгортання дає користувачу повний контроль над платформою .NET Core, що використовується його застосунком. Особливі характеристики усіх типів розгортання дають гарантії того що користувач зможе розробити застосунок найкращим способом в залежності від його потреб.

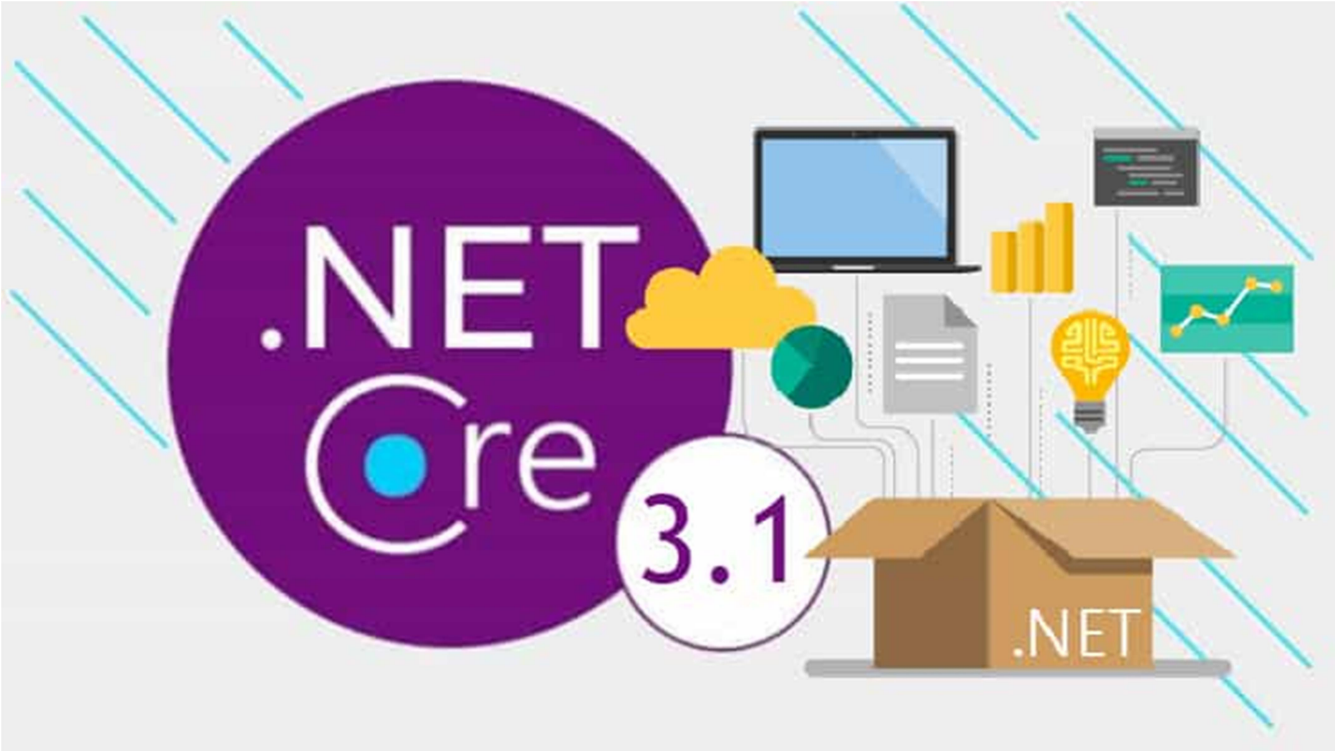

Рисунок 3.2 — Логотип .NET Core

Ця корисна програмна платформа тривалий час працює на Windows, Mac OSX а також Linux. Хоч це і здається дивним кроком для Microsoft, однак це вкрай важливо в комп'ютерному світі, який з кожним днем все більше орієнтований на гнучкість і структурування, коли справа доходить до ОС і платформ.

Архітектура .NET підтримує три мови програмування.

-C# - (вимовляється як "сі шарп") - сучасна об'єктно-орієнтована мова програмування. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript.

-F# - мова F# підтримує функціональні, об'єктно-орієнтовані і імперативні моделі програмування.

- Visual Basic - серед мов .NET синтаксис Visual Basic краще всього відповідає звичайній природній мові, що значно спрощує його вивчення. На

відміну від C# і F#, для яких корпорація Майкрософт активно розробляє нові функції, мова Visual Basic є стабільною. Visual Basic не підтримується для веб-застосувань, але підтримується для веб-API.

3.3. Message Passing Interface

Це специфікація, яку розробила у 1993—1994 роках група MPI Forum. Вона реалізує модель обміну повідомленнями між кількома процесами. Найсвіжіша версія специфікації це MPI-2. Під час виконання MPI програма породжує декілька процесів, що взаємодіють один з одним способом виклику прийому й передачі повідомлень.

Як правило, при запуску MPI-програм створюється заданий набір процесів, до того ж кожний з них виконується на окремому процесорі. У цих процесах часом виконуються різні програми, отже MPI-модель іноді вважають MIMD-моделлю (Multiple Instruction, Multiple Data), на відміну від SIMD моделі, де на різних процесорах виконуються лише однакові завдання. MPI підтримує точкові і глобальні, з блокуванням і без нього, види зв’язків. Створений комунікатор ховає від користувача структури обміну повідомленнями. Структура комунікацій часом змінюється під час існування процесу, але кількість процесів незмінна. Технологія MPI забезпечує перетворення заданих програм на рівні вихідних кодів. Підтримує роботу на кластерних і мультипроцесорних комплексах. Не підтримує запуск додаткових процесів, коли MPI-програма вже запущена. У технології відсутні паралельні введення та виведення, і налагодження програм, проте ці функції можуть бути додані до складу MPI як плагіни чи утиліти. Важливою властивістю паралельних програм є детермінізм — програма зобов’язана давати один і той самий результат для однакових наборів вхідних даних незалежно від кількості процесів. Технологія передачі даних загалом такої властивості не має, оскільки не визначено правила

отримування повідомлень від кількох процесів одним. Якщо ж один з процесів почергово надсилає декілька повідомлень іншому, MPI створить умови, щоб одержувач отримав їх у вказаному порядку. Відповідальність за забезпечення детермінованого виконання програми покладається на користувача.

4. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМИ ГЕНЕРАЦІЇ ГІДРОАКУСТИЧНОГО СИГНАЛУ

Програмний продукт для генерації гідроакустичного сигналу був написаний за допомогою мови програмування C#, у середовищі розробки програмного забезпечення Visual Studio 2019, з використанням методів паралельного програмування MPI.

Програмний продукт розраховує моментальні значення коливальних швидкостей та сумарного тиску для представлення акустичного поля сигналу у вигляді графіка звукових хвиль.

4.1. Налаштування середовища MPI

Перш за все, необхідно об`єднати комп`ютери, створивши локальну мережу, і командою ping переконатися, що комп'ютери мають доступ один до одного.

Для коректної роботи будь-якої MPI-програми необхідно встановити msmpisdk.exe і msmpisetup.exe, що доступні для завантажування на сайті Miscrosoft.

Після їх встановлення, у вказаній директорії з`являться:

mpiexec.exe — програма, яка дозволяє запускати MPI-програми з певними параметрами.

smpd.exe — програма, яка встановлює зв`язок комп`ютерів у мережі, і відображає роботу методів MPI на кожному з них (рис 4.1).

Протягом усієї роботи MPI-програми, smpd.exe має бути запущеним в окремій консолі на всіх комп`ютерах, що приймають участь у роботі.

Для зручності запуску шлях до них можна додати до складу змінної середовища PATH.

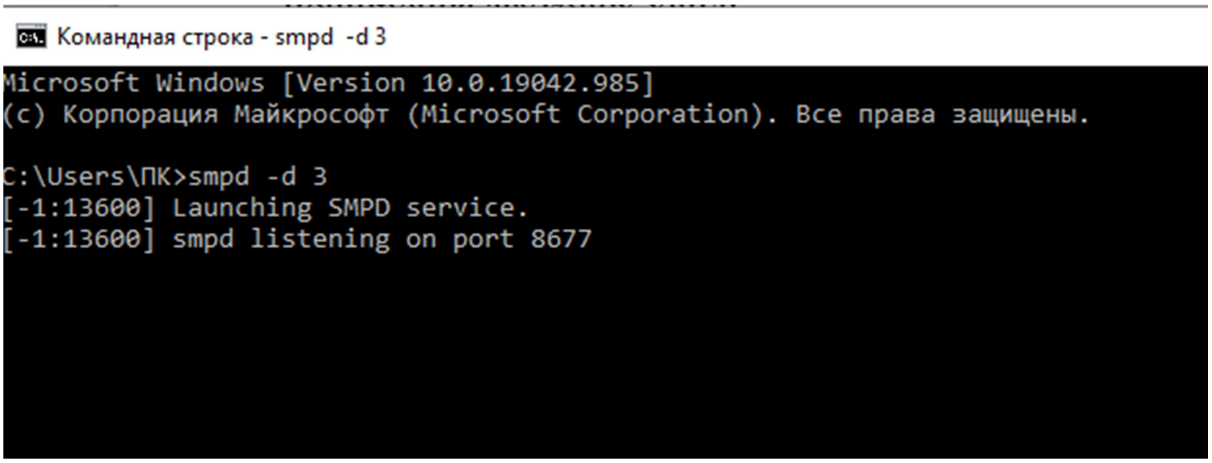

Рисунок 4.1 — smpd.exe запущений і очікує MPI-команд

Також, MPI вимагає, щоб усі комп`ютери у мережі мали однакові імена користувача та пароль, інакше виникне помилка доступу при спробі запустити програму на віддаленому комп`ютері.

4.2. Робота з програмою

Перш за все, користувач має запустити smpd.exe з параметрами -d 3, як вказано на рисунку 4.1, в окремому вікні консолі на усіх комп`ютерах, що прийматимуть участь у виконанні паралельної програми.

Для запуску будь-якої MPI-програми необхідний mpiexec.exe, про який було сказано в попередньому пункті. Його необхідно запустити в іншому вікні консолі, з параметрами -n ***кількість процесів,*** -machinefile ***шлях до файлу з IP-адресами комп`ютерів*** і ***шлях до .exe-файлу виконуваної програми*** (рис. 4.2).

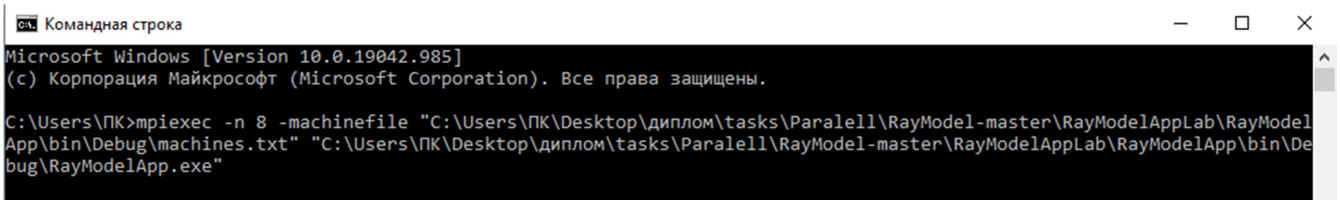

Рисунок 4.2 — Запуск програми за допомогою mpiexec.exe на 8 процесорах

Поки йде запуск програми на декількох процесах, на кожному комп`ютері у вікні з smpd.exe будуть відображатися дії, пов`язані з MPI, що були виконані на цьому комп`ютері (рис 4.3).

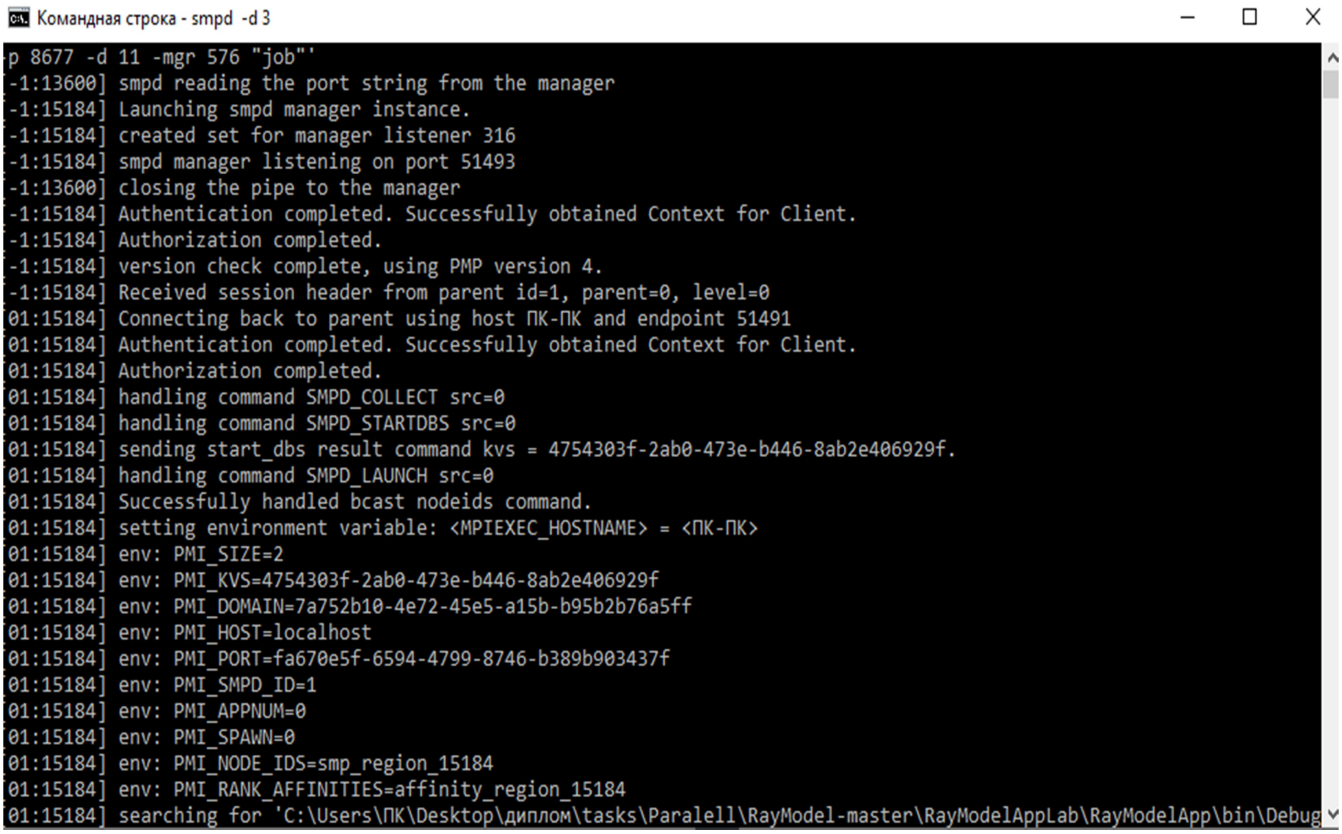

Рисунок 4.3 — smpd.exe у роботі

Як тільки mpiexec запустить програму, відкриється головне вікно (рис. 4.4).

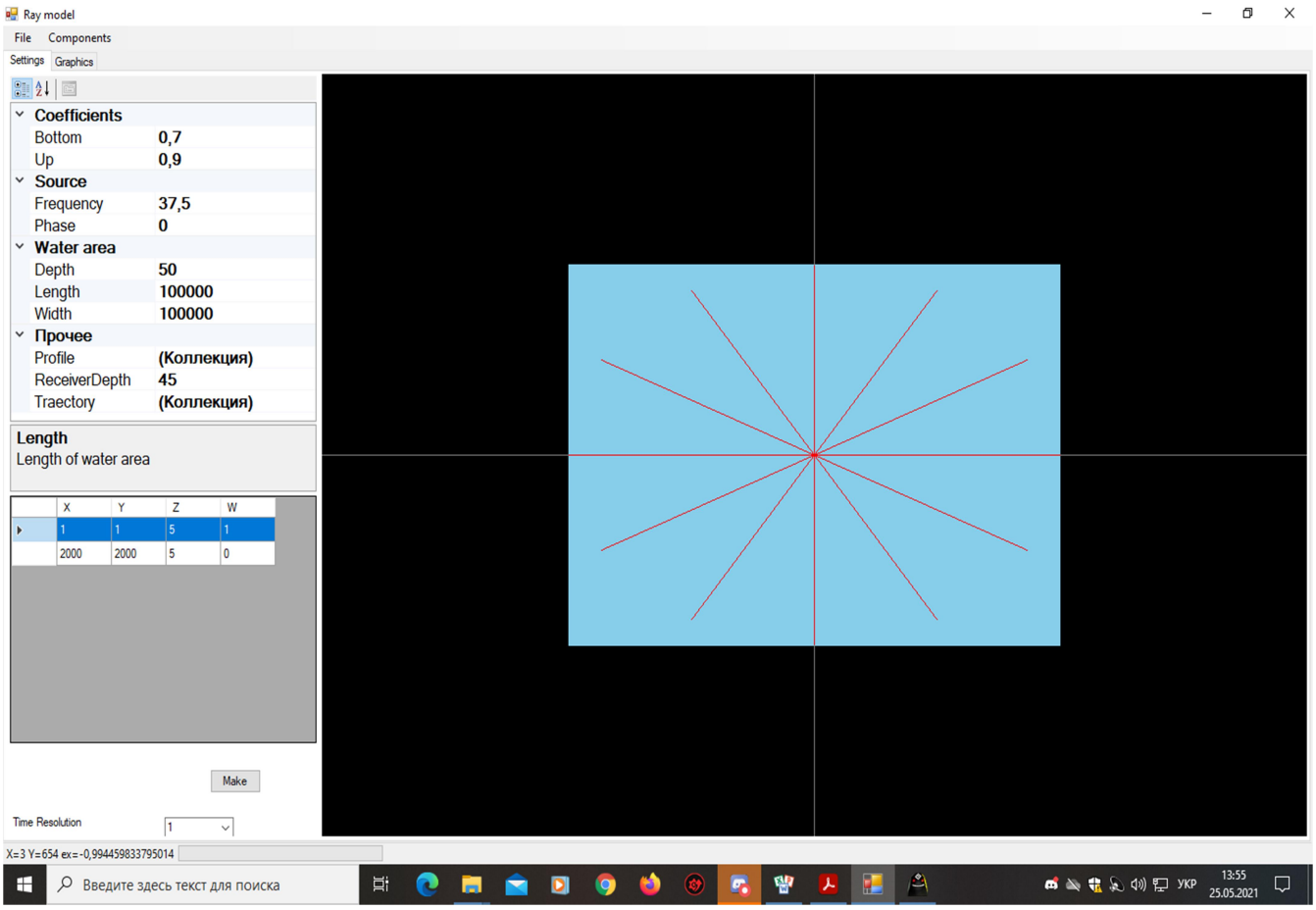

Рисунок 4.4 — Головне вікно програми

Перед запуском програми можна змінити вхідні дані, вказавши частоту, фазу, розміри водойми тощо (рис 4.5).

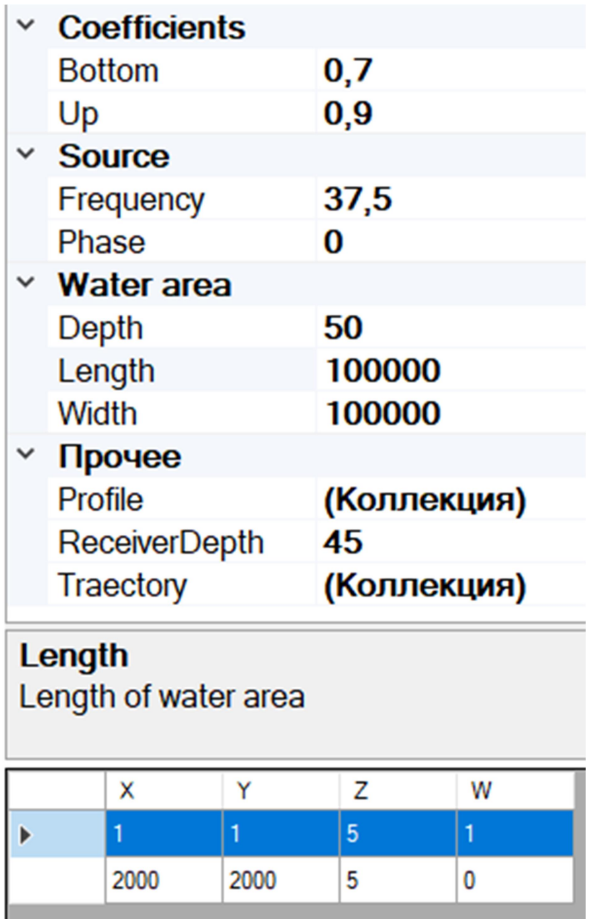

Рисунок 4.5 — введення вхідних даних

Після того, як дані введені, необхідно натиснути кнопку Make.

Далі, на екрані поруч з Make з`явиться кнопка RUN (рис 4.6).

Рисунок 4.6 — Кнопка RUN — запускає процес генерації сигналу

Під час роботи, подивившись через диспетчер задач, можна переконатися, що задіяні усі 8 ядер процесора (рис 4.7).

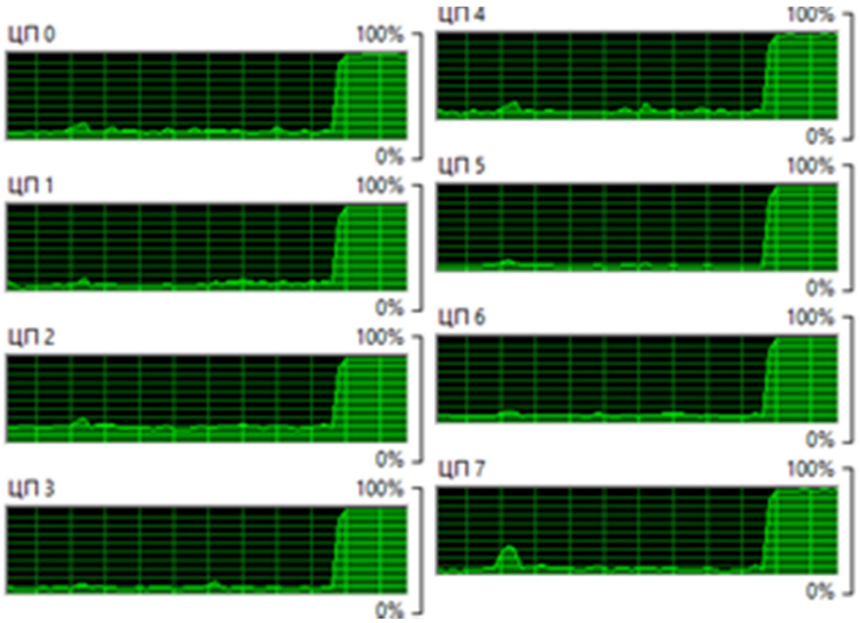

Рисунок 4.7 — Усі 8 процесорів завантажені роботою

Після завершення роботи програми, результат можна побачити, натиснувши Graphics (рис. 4.8).

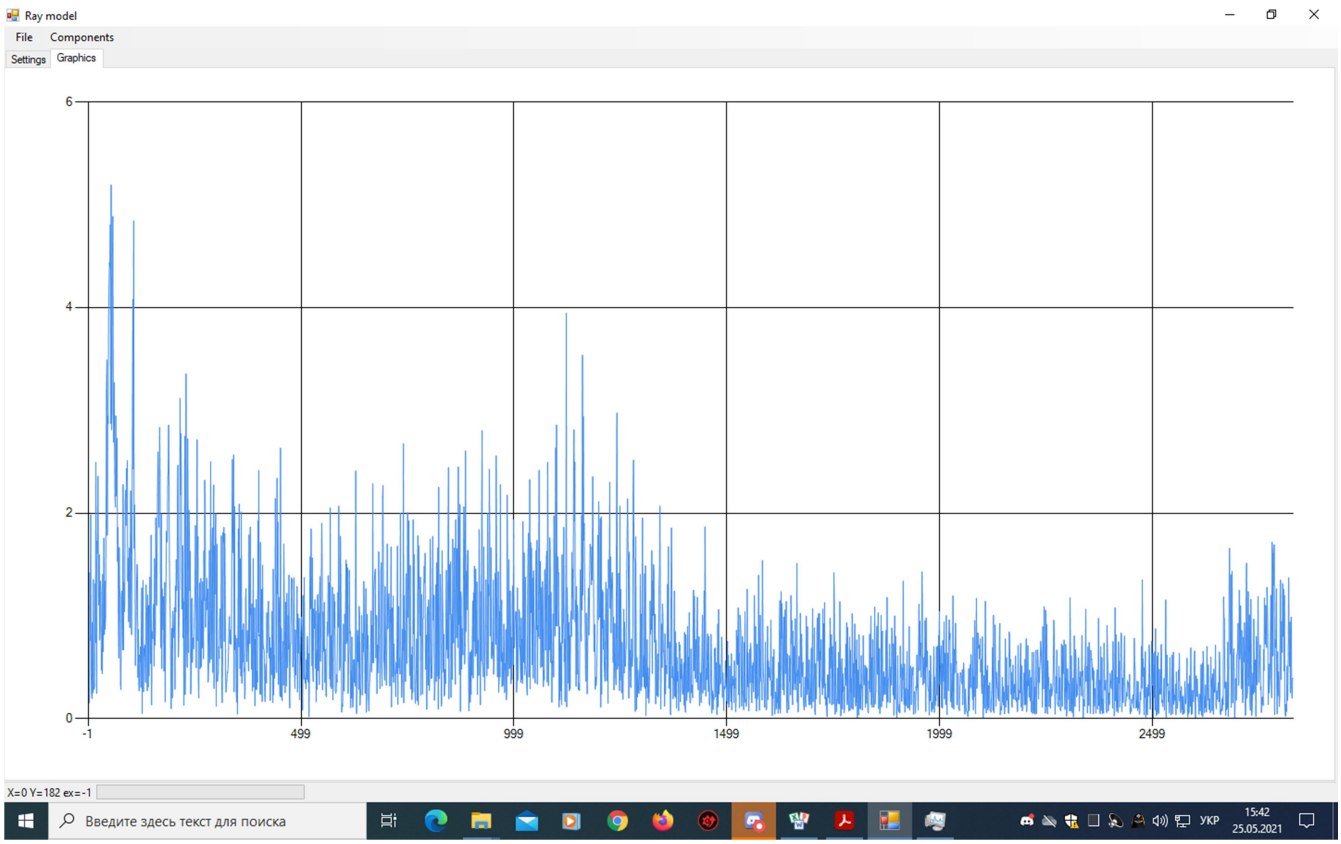

Рисунок 4.8 — Результат генерації гідроакустичного сигналу

ВИСНОВКИ

Під час проходження практики було створено застосунок, що містить для моделювання гідроакустичних сигналів. Програмний продукт написано мовою C# та на платформах .NET Core з використанням паралельного обчислення MPI.

Створені функціональні можливості дозволяють виконувати задачі — моделювання гідроакустичного сигналу і представлення його у вигляді графіку.

Користувач має змогу власноруч задавати параметри вхідних даних. На виході генерується єдиний сигнал, який і є результатом виконання генерації, та представлений у вигляді графіку.

Створена система істотно спрощує процес моделювання гідроакустичних сигналів та дозволяє досягти чіткого результату в межах більш складних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Доклады XII научной школы-семинара им. акад. Л.М.Бреховских "Акустика океана",М.: ГЕОС, 2009 — 486 с.

2. Hovem, J. M. (2010). Marine Acoustics: The Physics of Sound in Underwater Environments. Peninsula Publishing, Los Altos, California, USA. — 97 с.

3. Francois, R. E., and G. R. Garrison. 1982. Sound absorption based on ocean measurements, 72(6), 1879—1890. — 137 с.

5. Алексеев Г.В., Комашинская Т.С. Об активной минимизации потенциальной энергии звукового поля в двумерном многомодовом волноводе // Акустический журнал. — 2003. — Т. 49, №2. — С. 149—155.

7. Подласов Е.С. О моделировании волновых процессов в рамках лучевого приближения, // Подводная акустика. — 2005 — 58 с.

9. Келлер Дж. П. Распространение волн и подводная акустика / Пападакис Дж. Келлер Дж.., 1980 — 280 с.

10. Microsoft Visual Studio [Електронний ресурс] — Режим доступу до ресурсу: https://www.visualstudio.com/.

11. Rodriquez O.C. Traceo Ray tracing program [Електронний ресурс] / Rodriquez O.C.. — 2011. — Режим доступу до ресурсу: www.siplab.fct.ualg.pt.

12. Modeling and simulation of hydro-acoustic signal propagation in the Romanian Black Sea coast / E.G. Curcă, R.G. Damian, O. Radu, V. Roçca. — Research Center for Navy, Çtefănită Vodă Street, N, 2015. — 29 с.

13. Arctan - Powerful Acoustic Simulation Software [Електронний ресурс] — Режим доступу до ресурсу: https://www.mscsoftware.com/.

14. Fox D N. The modular Ocean Data Assimilation System Oceanography / Fox D N., 2002. — 93 с.

15. Leroy C C. A new equation for the accurate calculation of sound speed in all oceans / Leroy C C. — Acoust. Soc. Am, 2008. — 124 с.

17. Авилов К.В., Добряков Н.А., Попов О.Е. Комплекс программных средств для вычисления звуковых полей в морской среде, неоднородной по глубине и трассе распространения // Доклады X научной школы-семинара академика Л. М. Бреховских "Акустика океана", совмещенной с XIV сессией Российского акустического общества. — М.: ГЕОС, 2004. — С. 27—30.

18. Гусев В. Г. Системы пространственно-временной обработки гидроакустической информации / В. Г. Гусев. — Л.: Судостроение, 1988. — 264

19. Справочник по гидроакустике / А. П. Евтютов, А. Е. Колесников, Е. А. Корепин и др. — 2-е изд., перераб. и доп. — Л.: Судостроение, 1988. — 552 с.

20. Воеводин В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. — СПб.: БХВ-Петербург, 2002. — 608 с.

21. Антонов А. С. Технологии параллельного программирования MPI и OpenMP: Учеб. пособие / А. С. Антонов. — М.: Изд-во МГУ, 2012. — 344 с.

ДОДАТОК А

Модуль генерації гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI

Специфікація

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
	ІвановВ_Записка	Пояснювальна записка
	Form1.cs	Інтерфейс системи
	RayModelApp.cs	Модуль обрахунку сигналу
	FrmTraectories.cs	Модуль обрахунку траєкторії руху сигналу
	Sreda.cs	Модуль обробки характеристик водойми

ДОДАТОК Б

Модуль генерації гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI

Лістинг програмного продукту

Аркушів 8

Київ - 2021

```
using System;
using MPI;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using OpenTK;
using OpenTK.Graphics.OpenGL;
using System.Windows.Media.Media3D;
using System.Reflection;
using System.IO;
using System.Diagnostics;
using System.Collections.Concurrent;
using System.Threading.Tasks;
using mc3vray;

namespace RayModelApp
{
    public partial class RayForm : Form
    {
        #region Global Var

        Sreda Sr = new Sreda();
        BindingList<Point4D> Points = new BindingList<Point4D>();
        List<Point3D> GraphPoints = new List<Point3D>();
        static int pointLength;
        static ConcurrentQueue<Point4D> queue = new ConcurrentQueue<Point4D>();
        static List<Pnt> pnts = new List<Pnt>();
        static object locker = new object();
        Assembly asm;
        Type type;
        static MethodInfo method;
        static object obj;
        static double Omega;
        static int ErrPoint;

        double[] hobj = null;
        double[] lobj = null;
        double[] time = null;

        static double stFreq;
        static double stGAS;

        bool loaded = false;
        bool lbDown = false;

        float x = 0;
        float y = 0;
        float z = -1f;

        double kx = 1;
        double ky = 1;

        double glX = 0;
        double glY = 0;
        double downX = 0;
```

```
int nxGrid = 3;
int nyGrid = 4;

string sProfileDir = "";

#endregion

//

#region Load

public RayForm()
{
    InitializeComponent();

    #region LoadRayModel

    asm = Assembly.LoadFrom("mc3vray.dll");
    type = asm.GetType("mc3vray.GObject", true, true);
    obj = Activator.CreateInstance(type);
    method = type.GetMethod("calcAmp");

    #endregion


    ofd.InitialDirectory = System.Environment.CurrentDirectory;
    ofd.Filter = "Environment file|*.env|Trajectory file|*.tr";

    sfd.InitialDirectory = ofd.InitialDirectory;
    sfd.Filter = ofd.Filter;


    chart1.ChartAreas[0].CursorX.IsUserEnabled = true;
    chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;
    chart1.ChartAreas[0].CursorY.IsUserEnabled = true;
    chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;
    chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
    chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
}

private void RayForm_Load(object sender, EventArgs e)
{
    #region Sr

    Sr.Length = 100000;
    Sr.Width = 100000;
    Sr.Depth = 50;

    Sr.Up = 0.9; Ray.Ksrf = 0.9;
    Sr.Bottom = 0.7; Ray.Kbtm = 0.7;

    Sr.Frequency = 37.5f;
    Sr.Phase = 0.0f;
    Sr.ReceiverDepth = 45; Ray.Hgas = 45;

    Sr.Traectory.Add(new Point3D() { X = 1, Y = 1, Z = 5 });
    Sr.Traectory.Add(new Point3D() { X = 2000, Y = 2000, Z = 5 });
```

```
    Sr.Profile.Add(new ProfilePoint() { z = 10, c = 1542, p = 25.1f, t = 19.0f });
    Sr.Profile.Add(new ProfilePoint() { z = 30, c = 1545, p = 27.3f, t = 15.0f });
    Sr.Profile.Add(new ProfilePoint() { z = 50, c = 1548, p = 29.3f, t = 10.0f });

    Points.Add(new Point4D() { X = 1, Y = 1, Z = 5, W = 1 });
    Points.Add(new Point4D() { X = 2000, Y = 2000, Z = 5, W = 0 });

    pg1.SelectedObject = Sr;
    pg1.PropertySort = PropertySort.Categorized;


    Ray.Hz.Clear();
    Ray.Hz.AddRange(Sr.Profile.Select(p => (double)p.z).ToArray());

    Ray.Cz.Clear();
    Ray.Cz.AddRange(Sr.Profile.Select(p => (double)p.c).ToArray());

    #endregion

    #region dgw

    GLC.MouseWheel += OnMouseWheel;

    dgw.AutoGenerateColumns = false;
    dgw.Columns.Add("X", "X");
    dgw.Columns.Add("Y", "Y");
    dgw.Columns.Add("Z", "Z");
    dgw.Columns.Add("V", "W");
    dgw.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dgw.AllowUserToAddRows = true;

    foreach (DataGridViewTextBoxColumn c in dgw.Columns)
        c.Width = 60;

    dgw.Columns["X"].DataPropertyName = "X";
    dgw.Columns["Y"].DataPropertyName = "Y";
    dgw.Columns["Z"].DataPropertyName = "Z";
    dgw.Columns["V"].DataPropertyName = "W";

    dgw.DataSource = Points;

    #endregion

    bArtRun.Visible = false;
}

#endregion

//

#region Set/Load Parametr

private void button1_Click(object sender, EventArgs e)
{
    stCoord.Text = string.Format("Capacity {0}", GraphPoints.Count);
    bArtRun.Visible = false;
```

```
 private void cmiTraectories_Click(object sender, EventArgs e)
        {
            using (FrmTraectories f = new FrmTraectories())
            {
                if (f.ShowDialog() == DialogResult.OK)
                {
                    dgw.DataSource = null;
                    Points.Clear();
                    Sr.Traectory.Clear();
                    foreach (var p in f.points)
                    {
                        Sr.Traectory.Add(new Point3D() { X = p.X, Y = p.Y, Z = p.Z });
                        Points.Add(new Point4D() { X = p.X, Y = p.Y, Z = p.Z, W = 0 });
                    }
                    dgw.DataSource = Points;

                    bArtRun.Visible = false;
                }
            }
        }

        private void cmiProfiles_Click(object sender, EventArgs e)
        {
            using (FrmProfiles f = new FrmProfiles())
            {
                if (f.ShowDialog() == DialogResult.OK)
                {
                    Sr.Profile.Clear();
                    foreach (var p in f.p.Points)
                        Sr.Profile.Add(p);

                    bArtRun.Visible = false;
                }
            }
        }


        private void SaveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (sfd.ShowDialog() == DialogResult.OK)
            {
                switch (sfd.FilterIndex)
                {
                    case 0:
                        using (TextWriter tw = File.CreateText(sfd.FileName))
                        {
                            foreach (Point4D p in Points)
                                tw.WriteLine(string.Format("{0}\t{1}\t{2}\t{3}", p.X, p.Y,
p.Z, p.W));
                        }
                        break;
                    case 1:
                        Sr.Save(sfd.FileName);
                        break;
                }
            }
        }
```

```
        {
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                switch (ofd.FilterIndex)
                {
                    case 0:
                        Points.Clear();
                        using (TextReader tr = File.OpenText(ofd.FileName))
                        {
                            string line;
                            double px, py, pz, pw;
                            while ((line = tr.ReadLine()) != null)
                            {
                                Console.WriteLine(line);
                                string[] vals = line.Split('\t');
                                double.TryParse(vals[0], out px);
                                double.TryParse(vals[1], out py);
                                double.TryParse(vals[2], out pz);
                                double.TryParse(vals[3], out pw);
                                Points.Add(new Point4D() { X = px, Y = py, Z = pz, W = pw
});
                            }
                        }
                        bArtRun.Visible = false;
                        break;
                    case 1:
                        Sr.Load(ofd.FileName);
                        pg1.SelectedObject = Sr;
                        bArtRun.Visible = false;
                        break;
                }
                GLC.Invalidate();
            }
        }

        #endregion

private void btnMakePoints_Click_1(object sender, EventArgs e)
        {
            Point4D p4d;
            List<double> times = new List<double>();
            double currentTime = 0;
            double dt, path, truepath, lastpath, lasttime, firstpath;
            double Lx, Ly, Lz;
            double px, py, pz;
            double dx, dy, dz;
            int nP;
            double.TryParse(cbTR.Text, out dt);
            GraphPoints.Clear();
            while (!queue.IsEmpty)
                queue.TryDequeue(out p4d);
            Console.WriteLine(string.Format("queue.Count {0}", queue.Count));
            lasttime = 0;
            firstpath = 0;
            px = Points[0].X;
            py = Points[0].Y;
            pz = Points[0].Z;
            GraphPoints.Add(new Point3D() { X = px, Y = py, Z = pz })
```

```
Console.WriteLine(string.Format("Точка[{0}] {1} {2}", GraphPoints.Count, px, py));
            for (int i = 0; i < Points.Count - 1; i++)
            {
                Console.WriteLine();
                path = getDist(Points[i], Points[i + 1]);
                Lx = Points[i + 1].X - Points[i].X;
                Ly = Points[i + 1].Y - Points[i].Y;
                Lz = Points[i + 1].Z - Points[i].Z;

                dx = Points[i].W * Lx / Math.Sqrt(Lx * Lx + Ly * Ly + Lz * Lz);
                dy = Points[i].W * Ly / Math.Sqrt(Lx * Lx + Ly * Ly + Lz * Lz);
                dz = Points[i].W * Lz / Math.Sqrt(Lx * Lx + Ly * Ly + Lz * Lz);
                Console.WriteLine(string.Format("dx {0} dy {1}", dx, dy));

                if (lasttime > 0)
                {
                    Console.WriteLine(string.Format("Начальное время {0}", dt - lasttime));
                    GraphPoints.RemoveAt(GraphPoints.Count - 1);
                    times.RemoveAt(times.Count - 1);
                    px += (dt - lasttime) * dx;
                    py += (dt - lasttime) * dy;
                    pz += (dt - lasttime) * dz;
                    currentTime += dt - lasttime;
                    GraphPoints.Add(new Point3D() { X = px, Y = py, Z = pz });
                    times.Add(currentTime);
                    Console.WriteLine(string.Format("Точка[{0}] {1} {2}", GraphPoints.Count,
px, py));
                    firstpath = Math.Sqrt(Math.Pow((dt - lasttime) * dx, 2) + Math.Pow((dt -
lasttime) * dy, 2) + Math.Pow((dt - lasttime) * dz, 2));
                    path -= firstpath;
                }
                nP = (int)Math.Floor(path / (Points[i].W * dt));
                truepath = nP * Points[i].W * dt;
                Console.WriteLine(string.Format("Отрезок {0} Длина {1} ЦелаяДлина {2}
Точок={3}", i, path, truepath, nP));
                for (int j = 0; j < nP; j++)
                {
                    px += dt * dx;
                    py += dt * dy;
                    pz += dt * dz;
                    currentTime += dt;
                    GraphPoints.Add(new Point3D() { X = px, Y = py, Z = pz });
                    times.Add(currentTime);
                    Console.WriteLine(string.Format("Точка[{0}] {1} {2}", GraphPoints.Count,
px, py));
                }
                lastpath = path - truepath;
                lasttime = lastpath / Points[i].W;
                Console.WriteLine(string.Format("ПоследняяДлина {0} ПоследнееВремя={1}",
lastpath, lasttime));
                if (lasttime > 0)
                {
                    px += lasttime * dx;
                    py += lasttime * dy;
                    pz += lasttime * dz;
                    currentTime += lasttime;
                    GraphPoints.Add(new Point3D() { X = px, Y = py, Z = pz });
                    times.Add(currentTime);
```

```
Console.WriteLine(string.Format("Точка[{0}] {1} {2}", GraphPoints.Count, px, py));
                }
            }
            GLC.Invalidate();
            for (int i = 0; i < GraphPoints.Count; i++)
                queue.Enqueue(new Point4D() { X = GraphPoints[i].X, Y = GraphPoints[i].Y, Z
= GraphPoints[i].Z, W = times[i] });
            times = null;
            Console.WriteLine(string.Format("GraphPoints.Count = {0} Queue ={1}",
GraphPoints.Count, queue.Count));
            pointLength = GraphPoints.Count;
            //TestPoint();

            Ray.Hz.Clear();
            Ray.Hz.AddRange(Sr.Profile.Select(p => (double)p.z).ToArray());

            Ray.Cz.Clear();
            Ray.Cz.AddRange(Sr.Profile.Select(p => (double)p.c).ToArray());

            bArtRun.Visible = true;
        }
```

ДОДАТОК В

Модуль генерації гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI

Опис програмного модуля

Аркушів 8

Київ - 2021

АНОТАЦІЯ

Застосунок надає структурний опис реалізації модуля для генерації гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI.

Програмне забезпечення надає можливість моделювання гідроакустичного сигналу з використанням потужностей декількох комп’ютерів при заданих параметрах акваторії.

Програмне забезпечення виконане мовою програмування C# у середовищі Visual Studio 2019 з застосуванням технології MPI.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код модуля міститься в межах класу RayModelMaster, що включає в себе методи для генерації гідроакустичного сигналу з застосуванням технології MPI. Модуль є ядром програмного продукту, адже виконує операції для моделювання сигналу. Необхідні дані про акваторію та сигнал сервіс приймає з класу Sreda.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмний продукт реалізує такі функціональні можливості:

—інтерфейс для отримання вхідних даних;

—моделювання гідроакустичного сигналу з застосуванням технології паралельних обчислень MPI;

—відображення отриманого результату на графіку.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ ПРОГРАМИ

Моделювання сигналу реалізовано у формі класу, який моделює сигнал методом уявних джерел з заданими параметрами та має такі методи:

1) btnMakePoints_Click_1();
2) OpenToolStripMenuItem_Click ();
3) SaveToolStripMenuItem_Click ();
4) RayForm_Load();
5) RayForm();
6) getDist();
7) bRun_Click();
8) btnSequential_Click();

-7-

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Програмний код модулю був виконаний за допомогою мови програмування C# у середовищі VisualStudio 2019 на базі платформи .NET Core.

Для запуску на декількох комп’ютерах було застосовано технологію паралельних обчислень MPI.

-8-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

— координати джерела сигналу;

— координати Гідроакустичної системи(ГАС);

— частота випромінювання джерела сигналу;

— час моделювання;

— глибина моря.

Вихідними даними є:

— відображений на графіку змодельований сигнал.