Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Оптимізація обчислень при моделюванні руху об'єктів з перешкодами за допомогою графічного процесору
(тема)

Виконав:
студент _2_ курсу, групи _САУм-19-1
Литвин І.Р.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз і управління
(повна назва освітньої програми)
Керівник доц. Гибкіна Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Тевяшев А.Д.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _124 Системний аналіз
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
"____" _____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Литвину Ігорю Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи _Оптимізація обчислень при моделюванні руху об'єктів з перешкодами за допомогою графічного процесору
затверджена наказом по університету від 23 жовтня 2020 р. № 1420 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2020 р.

3. Вихідні дані до роботи _Математична модель течії в'язкої нестисливої рідини в ізометричному середовищі

4. Перелік питань, що потрібно опрацювати в роботі
1. Системний аналіз проблеми дослідження течії в'язкої нестисливої рідини
2. Вибір і обґрунтування методу розв'язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз проблеми
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	вересень 2020 р.	виконано
2	Вибір та обґрунтування методу	жовтень – листопад 2020 р.	виконано
3	Розробка алгоритму і програми	листопад – грудень 2020 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	листопад – грудень 2020 р.	виконано
5	Робота над текстом пояснювальної записки	грудень 2020 р.	виконано
6	Представлення роботи на рецензію в ЕК	грудень 2020 р.	виконано

Дата видачі завдання _1_ вересня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Гибкіна Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 84 с., 12 табл., 26 рис., 1 дод., 12 джерел.

ГІДРОДИНАМІКА ЗГЛАДЖЕНИХ ЧАСТИНОК, СИСТЕМА РІВНЯНЬ НАВ'Є-СТОКСА, МЕТОД R-ФУНКЦІЙ, МЕТОД КРОКУЮЧИХ КУБИКІВ, МОДЕЛЮВАННЯ РІДИНИ, CUDA.

Об'єкт дослідження – течії в'язкої нестисливої рідини, що моделюються системою диференціальних рівнянь Нав'є-Стокса у формулюванні Лагранжа.

Мета роботи – оптимізувати процесс моделювання течії рідини з плином часу у тривимірному просторі за допомогою графічного процессору.

Методи дослідження – метод *R*-функцій, гідродинаміка згладжених частинок, метод крокуючих кубиків, CUDA.

В роботі оптимізований процес моделювання течії рідини з плином часу у тривимірному просторі на основі реалізації чисельних обчислень на графічному процесорі. Математичною моделлю є система диференціальних рівнянь Нав'є-Стокса у формулюванні Лагранжа, коли рідина представляється у вигляді скінченної кількості частинок, які взаємодіють між собою. Метод *R*-функцій використовується для побудови рівнянь меж тривимірних об'єктів, які потім будуть побудовані в області дослідження за допомогою алгоритму крокуючих кубиків.

За допомогою мови програмування C++ та графічної бібліотеки GLUT було реалізовано графічне представлення результатів обчислювальних експериментів, а чисельні обчислення були реалізовані на графічному процесорі за допомогою бібліотеки CUDA.

Отримані результати можуть бути використані у різних галузях науки і техніки, де необхідно провести математичне моделювання рідини в реальному часі з різними фізичними властивостями.

ABSTRACT

Introductory note: 84 pages, 12 tables, 26 figures, 1 appendixes, 12 sources.

R-FUNCTIONS METHOD, SMOOTHED PARTICLE HYDRODYNAMICS, NUMERICAL INTEGRATION, NAVIER-STOKES EQUATIONS, MARCHING CUBES METHOD, LIQUID SIMULATION, CUDA.

Object of research – flow of viscous incompressible fluid, described by the system of Navier-Stokes differential equations in the Lagrange formulation.

Aim of work – to optimize the behavior change of fluid flow in process time in three-dimensional space with help of graphical processor.

Research methods – *R*-functions method, smoothed particle hydrodynamics, marching cubes method, CUDA.

The paper optimizes the behavior of fluid flow in process of time in three-dimensional space on the basis of a combination of the method of R-functions and hydrodynamics of smoothed particles with the help of graphical processor. The mathematical model is the system of Navier-Stokes differential equations in the Lagrange formulation, where a liquid is represented as a discrete number of particles which interacting with each other. The *R*-functions method is used to construct equations for the boundaries of three-dimensional objects, which will be constructed in the field of research with the help of the marching cubes algorithm.

With the help of C++ object-oriented programming language and GLUT graphical library the graphical representation of the numerical experiments was implemented. The method of marching cubes allows us to build arbitrary objects, the equations of which are obtained using the method of R-functions, and numerical calculations were implemented on a graphics processor using the CUDA.

The obtained results can be used in various fields of science and technology, where it is necessary to carry out mathematical modeling of liquids in real time.

ЗМІСТ

ВСТУП

З моменту впровадження 3D-рендерингу на комп'ютерах, використання відеокарти перетворилося з простих пристроїв, призначених для виведення відео, на потужні паралельні обчислювальні пристрої. Обчислювальна потужність графічних процесорів (GPU) на сучасних відеокартах часто перевершує обчислювальну потужність центрального процесора, який ними керує. Однак до недавнього часу така обчислювальна потужність обмежувалась рендерингом складних 3D-сцен, що задовольняло потреби комп'ютерних геймерів та професійних дизайнерів. Зростаюча обчислювальна потужність графічних процесорів призвела до зростаючого інтересу до їх використання для обчислень за межі візуалізації відео; їх обчислювальна потужність сьогодні дозволяє перетворити настільний комп’ютер у високопродуктивний комп’ютер, здатний відповідати найдорожчим комп’ютерним кластерам за продуктивністю, але з невеликою вартістю, та бути використаним для наукових досліджень.

Високо паралельний характер графічних процесорів робить їх надзвичайно придатними інструментами для вдосконалених чисельних обчислювальних потреб, таких як наукове моделювання. Однак для повного використання їхніх можливостей потрібні відповідні інструменти та проблеми, які є обчислювально інтенсивними, а не залежать від кількості даних: усі дані не повинні міститися в пам'яті графічного процесора, а співвідношення операцій до даних має бути якомога більшим. Метод гідродинаміки згладжених частинок особливо підходить для обчислень на графічному процесорі, оскільки метод є обчислювально інтенсивним.

Реалізуючи усі обчислення на графічному процесорі, ми можемо отримати збільшення в 10 разів навіть для операція зв’заних з пам'яттю також додаткових переваг можна досягти, використовуючи спеціальні області кешованої пам'яті. Очевидно, що користь від операцій, пов'язаних з обчисленнями, набагато більша, враховуючи високопаралельний характер апаратного забезпечення GPU.

Метод гідродинаміки згладжених частинок широко використовується для моделювання рідини у різних галузях, але моделювання зазвичай потребує результатів які можна отримати в реальному часі, з цим може допомогти обчислювальна потужність графічних процесорів. Видобувна та енергетична галузі використовують відповідні моделі процесів (видобутку та транспортування) і об’єктів (турбін, компресорів,електрогенераторів, бойлерів, реакторів, парогенераторів і насосів) для запобігання аварій та несправностей і підвищення продуктивності роботи власних підприємств [1, 2]. Paul Cleary та ін. [3] використовували метод для моделювання литих систем в 3D. Геометрична складність литих систем призводить до того, що об'єкт важко зобразити як аналітичний вираз. Перша реалізація методу гідродинаміки згладжених частинок на GPU була реалізована Kolb та Cuntz та Harada [4]. Раніші роботи Amada та ін. [5] використовували графічний процесор лише для обчислення сили, тоді як адміністративні завдання, такі як пошук сусідів, виконувалися на центральному процесорі, вимагаючи не менше двох передач пам'яті хост-картки на кожній ітерації. І Kolb, і Cuntz, та Harada [6] використовували OpenGL і Cg (C forgraphics) для програмування графічного процесора, що вимагає знань комп’ютерної графіки, а також деяких трюків, щоб обійти обмеження, накладені цими мовами. Зокрема, оскільки OpenGL не надає можливості запису в тривимірні текстури, текстури положення та швидкості повинні бути приділені до двовимірної структури.

Таким чином, дана дипломна робота включає в себе оптимізацію методу на основі гідродинаміки згладжених частинок, шляхом виконня чисельних обчислень на графічному процессорі, високо паралельний характер якого робить їх надзвичайно придатними інструментами для вдосконалених чисельних обчислювальних потреб, таких як наукове моделювання.

1 СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ ТЕЧІЇ В'ЯЗКОЇ НЕСТИСЛИВОЇ РІДИНИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз проблеми дослідження течії в’язкої нестисливої рідини

1.1.1 Вербальна модель системи

Об’єктом аналізу є методи математичного моделювання течії в’язкої нестисливої рідини.

Предмет аналізу – вибір методів математичного моделювання течії в’язкої нестисливої рідини. Точкою зору є дослідник.

Мета – дослідити властивості чисельних методів та обрати найкращий.

Існує багато методів дослідження, які допомагають розв’язати систему диференціальних рівнянь Нав’є-Стокса: аналітичні та наближені. Чисельні методи поділяються на сіткові та наближено-аналітичні. Одним з наближено-аналітичних методів є метод гідродинаміки згладжених частинок. Основна ідея методу в тому що, рідина обробляється як набір дискретних елементів або вузлів, званих частинками. У кожної частинки є свої фізичні властивості (радіус, маса, густина, в’язкість) та просторова відстань – величина, на якій її властивості згладжуються функцією ядра. З цього слідує, що характеристики частинки можуть бути отримані шляхом сумуванням відповідних величин усіх частинок, які знаходяться в межі двох згладжених довжин, цей спосіб називають кубічним сплайном. Метод R-функцій дозволяє будувати межі будь-яких довільних геометричних об’єктів, тому знаючи координату частинки, можна легко визначити зіткнення частинки з межею геометричного об’єкта. Метод гідродинаміки згладжених частинок може застосовуватись в багатьох галузях досліджень, від розрахунків фізичних сил в астрофізиці та балістиці до розрахунку механіки деформованого твердого тіла для цивільного будівництва.

1.1.2 Морфологічний опис системи

Об’єктом дослідження є течія в’язкої нестисливої рідини. Описується дана модель системою нелінійних рівнянь Нав’є-Стокса.

Цільова функція системи – вектор швидкості.

Базисні функції системи – функція ядра, функції внутрішніх сил.

Додаткові функції системи – функції зовнішніх сил.

Формальна модель типу "чорний ящик" (рис. 1.1) є прямокутником, що означає систему. Він обмежений межами, стрілками зображено входи (вхідні величини) та виходи (вихідні величини) системи.

Входи – це те, що система використовує для своєї діяльності, з чим вона працює і що перетворює. Виходи – результат діяльності системи, те, що вона створює, в що перетворює вхідні величини відповідно до своїх функцій.

Зміст "чорного ящика" не розкривається, так як увага звертається тільки на межу системи. Межа, в свою чергу, підкреслює цілісність системи, відмежування її від зовнішнього середовища і взаємодія системи і середовища.

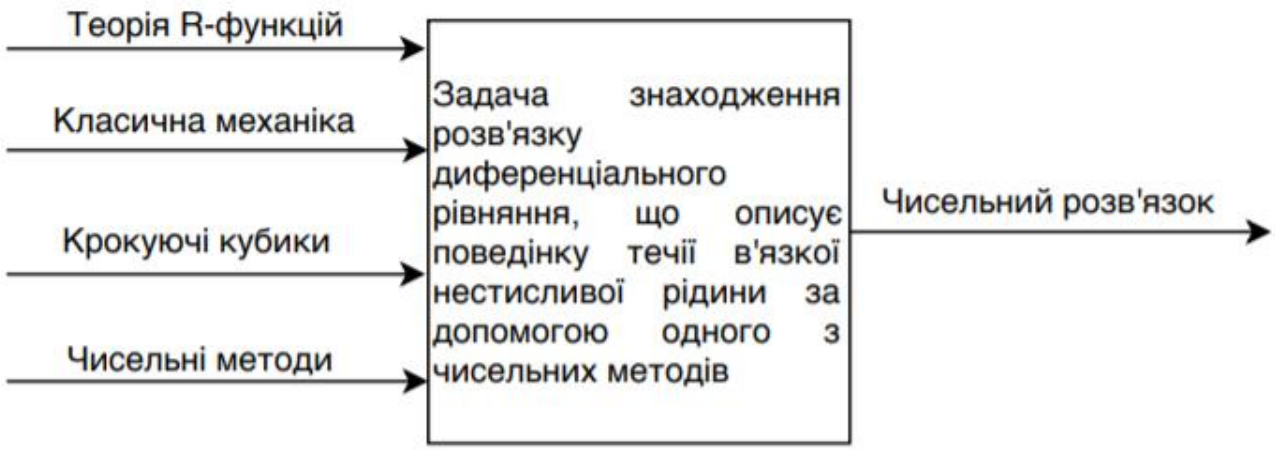


Рисунок 1.1 – Модель системи типу «чорний ящик»

1.1.3 Функціональна модель системи

В рамках методології IDEF0 модель системи описується за допомогою графічних IDEF0 діаграм і уточнюється за рахунок використання FEO, текстових і діаграм глосарію. При цьому модель включає в себе серію взаємопов'язаних діаграм, які поділяють складну систему на складові частини. Діаграми більш високого рівня (A-0, A0) – є найбільш загальним описом системи, представленим у вигляді окремих блоків. Декомпозиція цих блоків дозволяє досягати необхідного рівня деталізації опису системи. На рисунку 1.2 зображена IDEF0 діаграма – загальна функціональна модель.

Рисунок 1.2 – Контекстна IDEF0 діаграма

На рисунку 1.3 зображена декомпозиція діаграми 1.2 на 4 різних послідовно зв’язаних блоків. Найширшим та найскладнішим блоком є розв’язання диференціального рівняння, тому його потрібно розбити ще на 3 функціональні блоки з послідовними прямими зв’язками (рисунок 1.4).

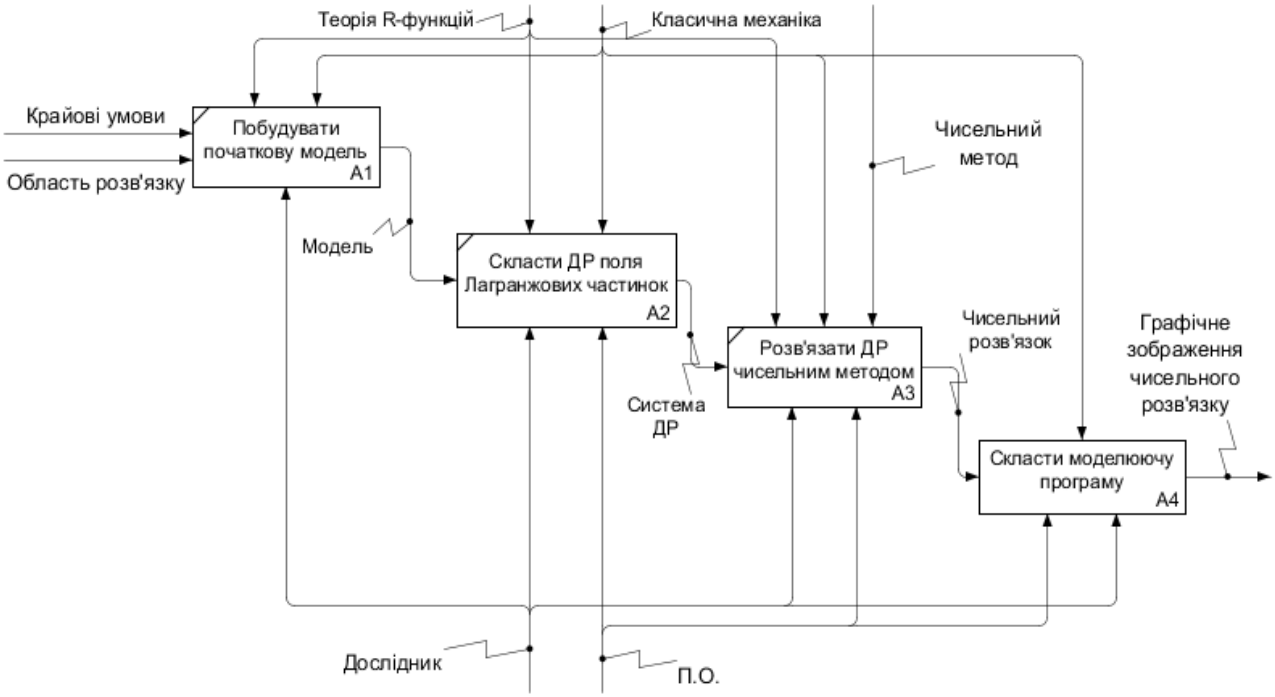


Рисунок 1.3 – Декомпозиція першого рівня функціонального блоку «Розв’язати системи рівнянь Нав’є-Стокса»

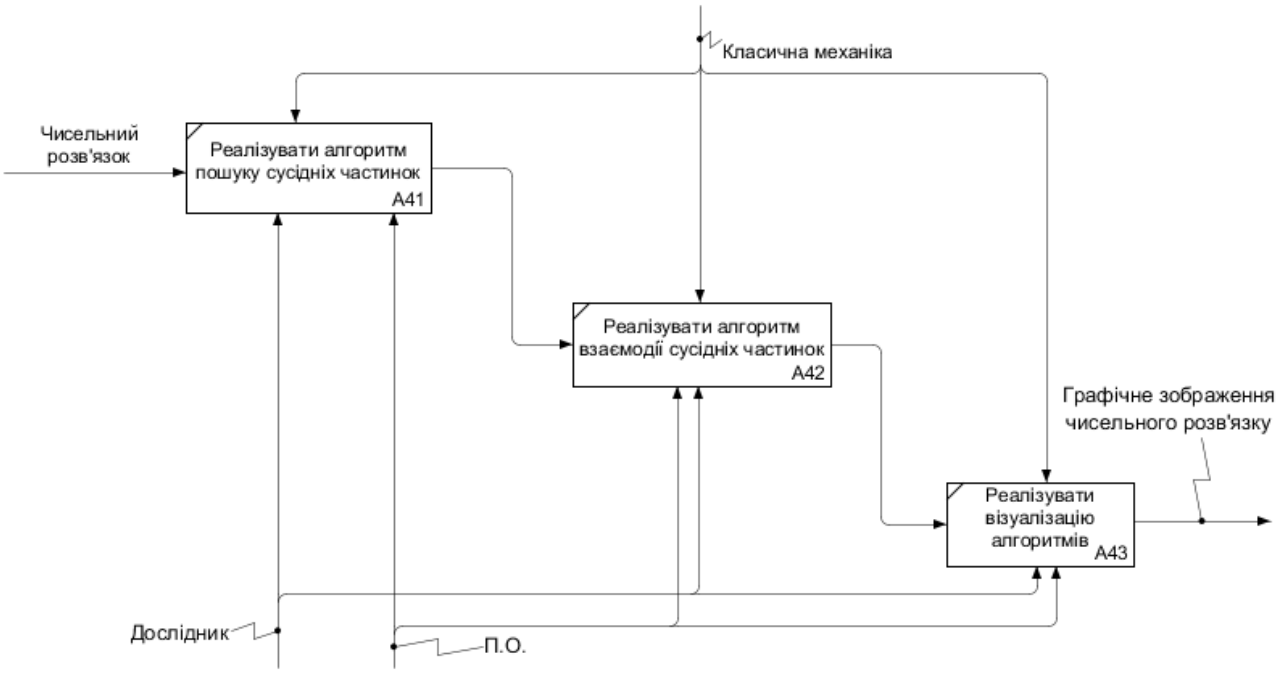


Рисунок 1.4 – Декомпозиція другого рівня функціонального блоку «Скласти моделюючу програму»

1.1.4 Інформаційна модель

Інформаційна модель системи також може бути представлена графічно за допомогою діаграми потоків даних DFD (рис. 1.5). Головна мета DFD – показати, як кожна робота перетворює свої вхідні дані у вихідні, а також виявити відносини між цими роботами. Аналогічно побудові функціональної моделі, після реалізації контекстної діаграми відбувається апроксимування системи до заданого рівня точності. Декомпозиція DFD зображена на рисунках 1.6-1.7.

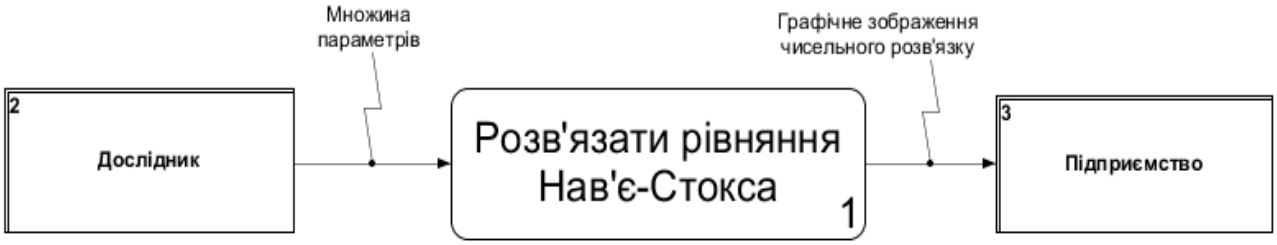


Рисунок 1.5 – Контекстна DFD діаграма

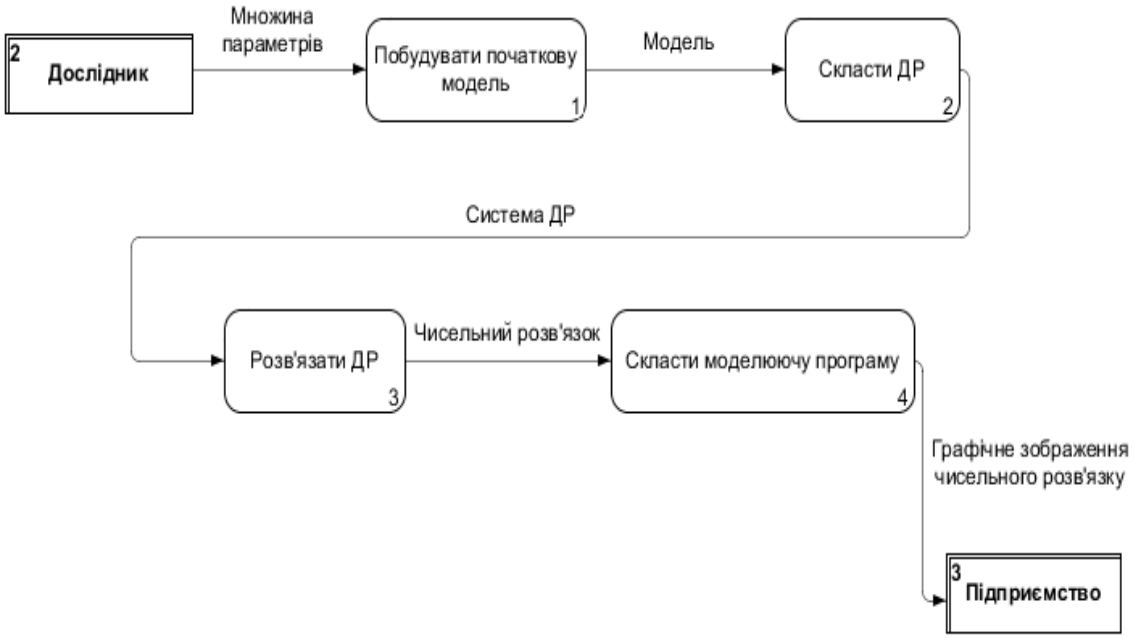


Рисунок 1.6 – Декомпозиція першого рівня DFD діаграми

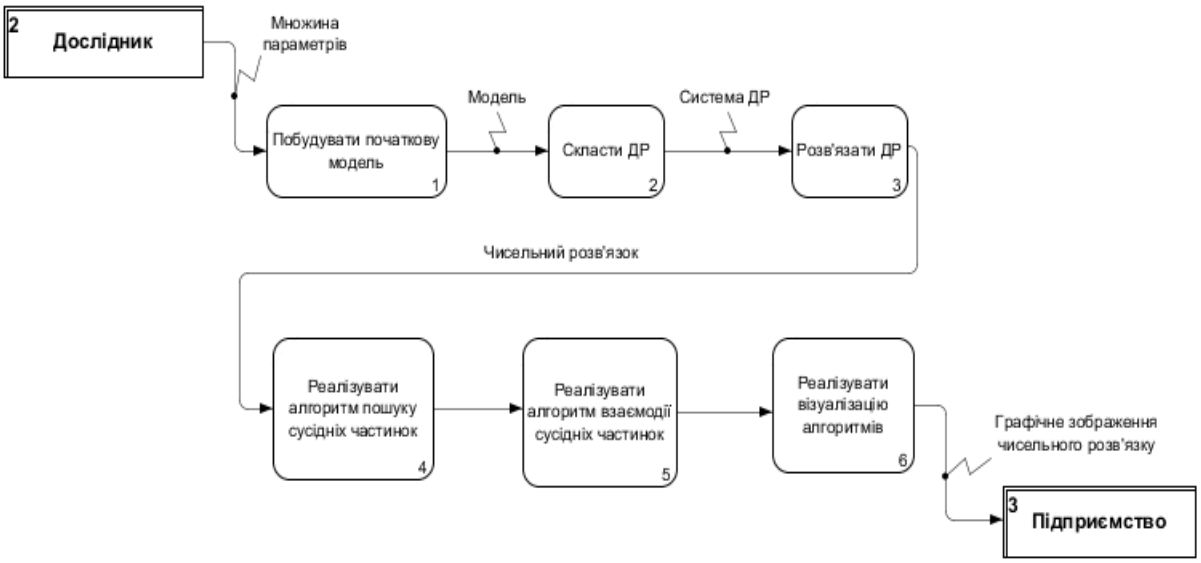


Рисунок 1.7 – Декомпозиція другого рівня DFD діаграми

За допомогою діаграми IDEF3 розглянемо послідовність операцій у невизначених процесах більш детально. Декомпозиція зображена на рисунку 1.8.

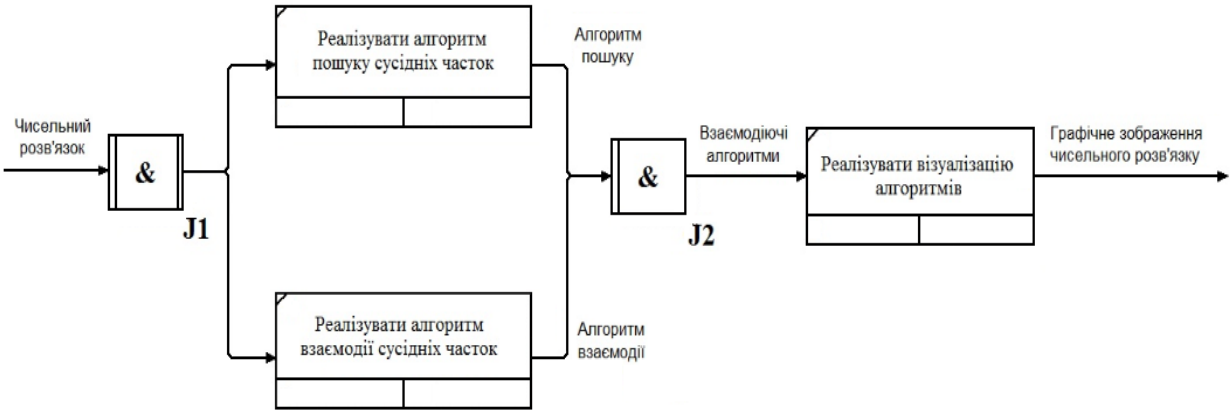


Рисунок 1.8 – Декомпозиція другого рівня IDEF3 діаграми

1.2 Аналіз сценаріїв вирішення проблеми дослідження течії в’язкої нестисливої рідини

1.2.1 Модель аналізу проблеми

Для того щоб провести аналіз проблеми дослідження течії в’язкої нестисливої рідини, потрібно розв’язати задачу вибору методу моделювання. Застосуємо метод аналізу ієрархій, за критерії візьмемо базові характеристики:

– точність розрахунків (К1);

– універсальність (К2);

– швидкість збіжності методу (К3);

– складність методу та його програмної реалізації (К4);

Множина, з якої буде прийняте рішення, складається із таких альтернатив:

– варіаційні методи (А1);

– точний розв’язок (А2);

– наближено-аналітичні методи (А3);

– сіткові методи (А4).

Таким чином, ієрархічна структура задачі складається із трьох рівнів:

– нульового – вибір методу розв’язання задачі дослідження течії в’язкої нестисливої рідини;

– першого – критерії порівняння методів;

– другого – множина альтернатив.

Скориставшись відповідною шкалою Т. Сааті побудуємо матрицю попарних порівнянь (табл. 1.1) для кожного рівня ієрархії. Ієрархічна структура задачі вибору методу має вигляд (рис 1.9). За допомогою цієї шкали одному порівняльному об’єкту можна поставити у відповідність деяке число ступеня переваги над іншим об’єктом. Знаючи ці данні стає можливим знайти вектори локальних пріоритетів, індекси та відношення узгодженості. Останні два допомагають визначити міру узгодженості результатів, отриманих експертним шляхом. Допустимим вважається значення відношення узгодженості $\leq 0,2$.

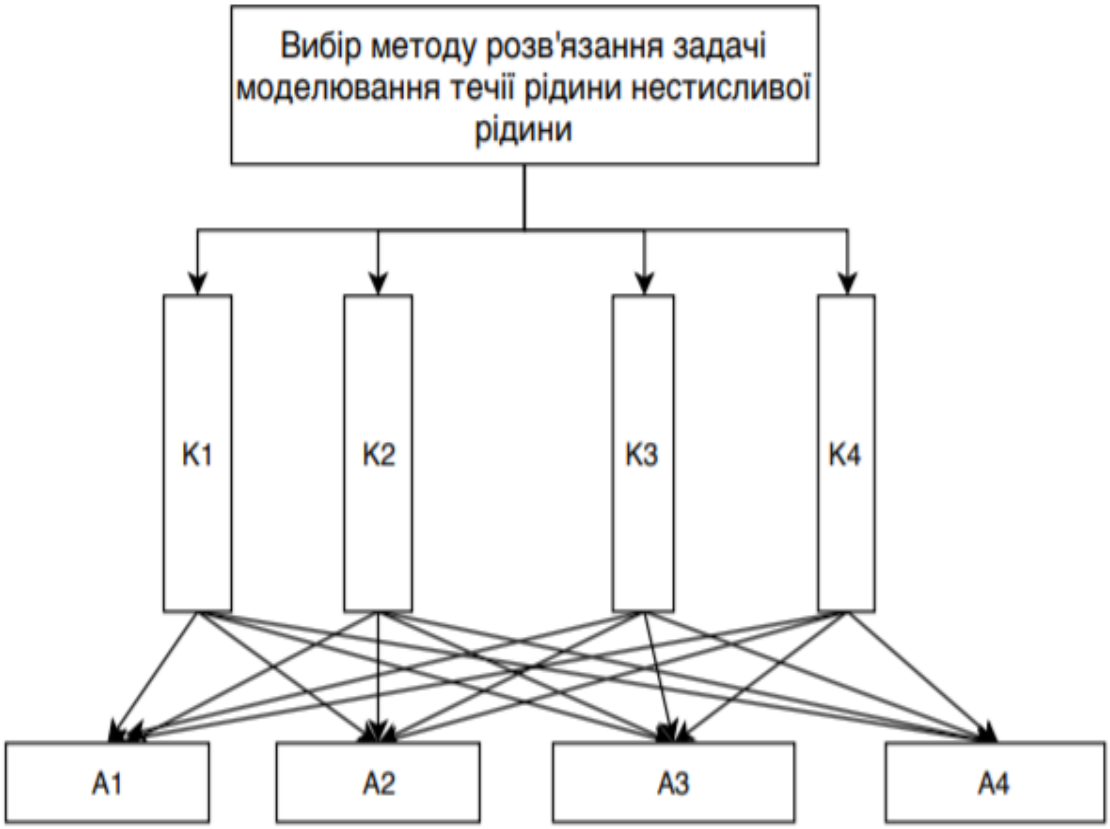


Рисунок 1.9 – Ієрархічна структура задачі вибору методу

Таблиця 1.1 – Матриця попарних порівнянь першого рівня

Номер п/п	К1	К2	К3	К4	Власний вектор	Вектор пріоритетів
К1	1	5	1/4	1	1,057	0,203
К2	1/5	1	1/5	1	0,447	0,086
К3	4	5	1	4	2,991	0,575
К4	1	1	1/4	1	0,707	0,136

Індекс узгодженості (ІУ) $= \frac{4,2204-4}{4-1} = 0,0735.$

Відносна узгодженість (ВУ) $= \frac{0,0735}{0,9} = 0,0816.$

З отриманих проміжних результатів можна зробити висновок, що матриця попарних порівнянь заповнена правильно, тобто жодне з тверджень не суперечить загальній логіці моделі, а дані узгоджені між собою на допустимому рівні. Тому вектор локальних пріоритетів відносно проблеми вибору набуває вигляду:

$$\vec{p}^{K} = (0.203,\ 0.086,\ 0.575,\ 0.136).$$

Аналогічним способом необхідно провести порівняльний аналіз альтернатив відносно кожного з критеріїв. Матриці попарних порівнянь представлені у вигляді таблиць 1.2 – 1.6.

Таблиця 1.2 – Матриця попарних порівнянь першого критерію

К1	А1	А2	А3	А4	Власний вектор	Вектор пріоритетів
А1	1	3	$1/2$	1	1,107	0,220
А2	$1/3$	1	$1/5$	1	0,508	0,101
А3	2	5	1	6	2,783	0,553
А4	1	1	$1/6$	1	0,693	0,127

Індекс узгодженості (ІУ) $= \dfrac{4{,}134 - 4}{4 - 1} = 0{,}0447$.

Відносна узгодженість (ВУ) $= \dfrac{0{,}0447}{0{,}9} = 0{,}0496$.

Вектор локальних пріоритетів: $\vec{p}_1^{A} = (0{,}220;\ 0{,}101;\ 0{,}553;\ 0{,}127)$.

Таблиця 1.3 – Матриця попарних порівнянь другого критерію

К2	А1	А2	А3	А4	Власний вектор	Вектор пріоритетів
А1	1	$1/7$	$1/7$	$1/7$	0,232	0,039
А2	7	1	1	5	2,432	0,412
А3	7	1	1	6	2,546	0,431
А4	7	$1/5$	$1/6$	1	0,695	0,118

Індекс узгодженості (ІУ) $= \dfrac{4,255-4}{4-1} = 0,0851.$

Відносна узгодженість (ВУ) $= \dfrac{0,0851}{0,9} = 0,0946.$

Вектор локальних пріоритетів: $\vec{p}_2^{\,A} = (0,039;\ 0,412;\ 0,431;\ 0,118).$

Таблиця 1.4 – Матриця попарних порівнянь третього критерію

К3	А1	А2	А3	А4	Власний вектор	Вектор пріоритетів
А1	1	8	2	8	3,364	0,542
А2	$1/8$	1	$1/8$	$1/4$	0,250	0,040
А3	$1/2$	8	1	4	2,000	0,322
А4	$1/8$	4	$1/4$	1	0,595	0,096

Індекс узгодженості (ІУ) $= \dfrac{4,150-4}{4-1} = 0,0500.$

Відносна узгодженість (ВУ) $= \frac{0,0500}{0,9} = 0,0556$.

Вектор локальних пріоритетів: $\vec{p}_3^A = (0,542;\ 0,040;\ 0,322;\ 0,096)$.

Таблиця 1.5 – Матриця попарних порівнянь четвертого критерію

К4	А1	А2	А3	А4	Власний вектор	Вектор пріоритетів
А1	1	2	1/8	1	0,707	0,110
А2	1/2	1	1/6	5	0,803	0,124
А3	8	6	1	9	4,559	0,706
А4	1	1/5	1/9	1	0,386	0,060

Індекс узгодженості (ІУ) $= \frac{4,243-4}{4-1} = 0,0809$.

Відносна узгодженість (ВУ) $= \frac{0,0809}{0,9} = 0,0899$.

Вектор локальних пріоритетів: $\vec{p}_4^A = (0,110;\ 0,124;\ 0,706;\ 0,060)$.

Таблиця 1.6 – Кінцеві результати задачі вибору методу розв’язання

Критерії / Альтернативи	К1	К2	К3	К4	Вектор пріоритетів
А1	0,220	0,039	0,542	0,110	0,374
А2	0,101	0,412	0,040	0,124	0,096
А3	0,553	0,431	0,322	0,706	0,431
А4	0,127	0,118	0,096	0,060	0,099

На основі отриманих результатів розрахуємо вектор глобальних пріоритетів та відповідні показники.

Індекс узгодженості (ІУ) $= 0{,}14544$.

Відносна узгодженість (ВУ) $= \dfrac{0{,}14544}{1{,}12 + 0{,}9} = 0{,}072$.

Вектор глобальних пріоритетів: $\vec{p} = (0{,}374;\ 0{,}096;\ 0{,}431;\ 0{,}099)$.

Як бачимо з таблиці 1.7, усі дані відповідають нормам узгодженості, а максимальна компонента вектора глобальних пріоритетів відповідає третій альтернативі, тобто метод, яким буде розв’язуватися задача, належить до наближено-аналітичних методів, а саме методу гідродинаміки згладжених частинок.

Визначившись із методом, перейдемо до моделі аналізу проблеми моделювання течії в’язкої нестисливої рідини. Для заданої системи головною незадоволеністю є максимізація ефективності моделювання. Вирішення проблеми почнемо із кластеризації властивостей системи, що відповідають певним характеристикам обраного методу моделювання. В результаті отримаємо три категорії незадоволень і відповідні характеристики для кожної з них:

а) бажані властивості;

1) збільшення точності результатів (Б1);

2) збільшення швидкодії програмної реалізації моделі (Б2);

3) вивіреність та достеменність розрахунків (Б3);

б) критичні властивості;

1) швидкодія програмної реалізації (К1);

2) багатоплатформність (К2);

в) небажані властивості;

1) отримання недостовірних результатів (Н1);

2) накопичення похибки розрахунків (Н2);

3) зменшення швидкодії програмної реалізації моделі (Н3).

Формалізуємо проблему у вигляді структури ієрархії (рисунок 1.10), виділивши в ній три рівня:

– нульовий – ціль моделі, що задається у вигляді проблеми незадоволеності;

– перший – класифікація незадоволень;

– другий – характеристики незадоволень у вигляді компонент проблеми.

Рисунок 1.10 – Ієрархічна структура аналізу проблеми

Таким чином, модель аналізу проблеми можна вважати описаною.

1.2.2 Оцінювання вектора пріоритетів незадоволеностей методом аналізу ієрархій

Продовжуючи аналіз проблеми, зокрема оцінювання глобального вектора пріоритетів, необхідно побудувати матрицю попарних порівнянь першого рівня, а також і кожної виділеної властивості. Отримані результати наведені у таблицях 1.7 – 1.10.

Таблиця 1.7 – Матриця попарних порівнянь першого рівня

Номер п/п	Бажані властивості	Критичні властивості	Небажані властивості	Власний вектор	Вектор пріоритетів
Бажані властивості	1	$\frac{1}{2}$	1	0,794	0,260
Критичні властивості	2	1	1	1,260	0,413
Небажані властивості	1	1	1	1,0	0,327

Таблиця 1.8 – Матриця попарних порівнянь бажаних властивостей

Бажані властивості	Збільшення точності результатів	Швидкодія програмної реалізації	Вивіреність розрахунків	Власний вектор	Вектор пріоритетів
Збільшення точності результатів	1	4	7	3,037	0,9963
Збільшення швидкодії програмної реалізації	$\frac{1}{4}$	1	2	0,187	1,0283
Вивіреність розрахунків	$\frac{1}{7}$	$\frac{1}{2}$	1	0,098	0,09774

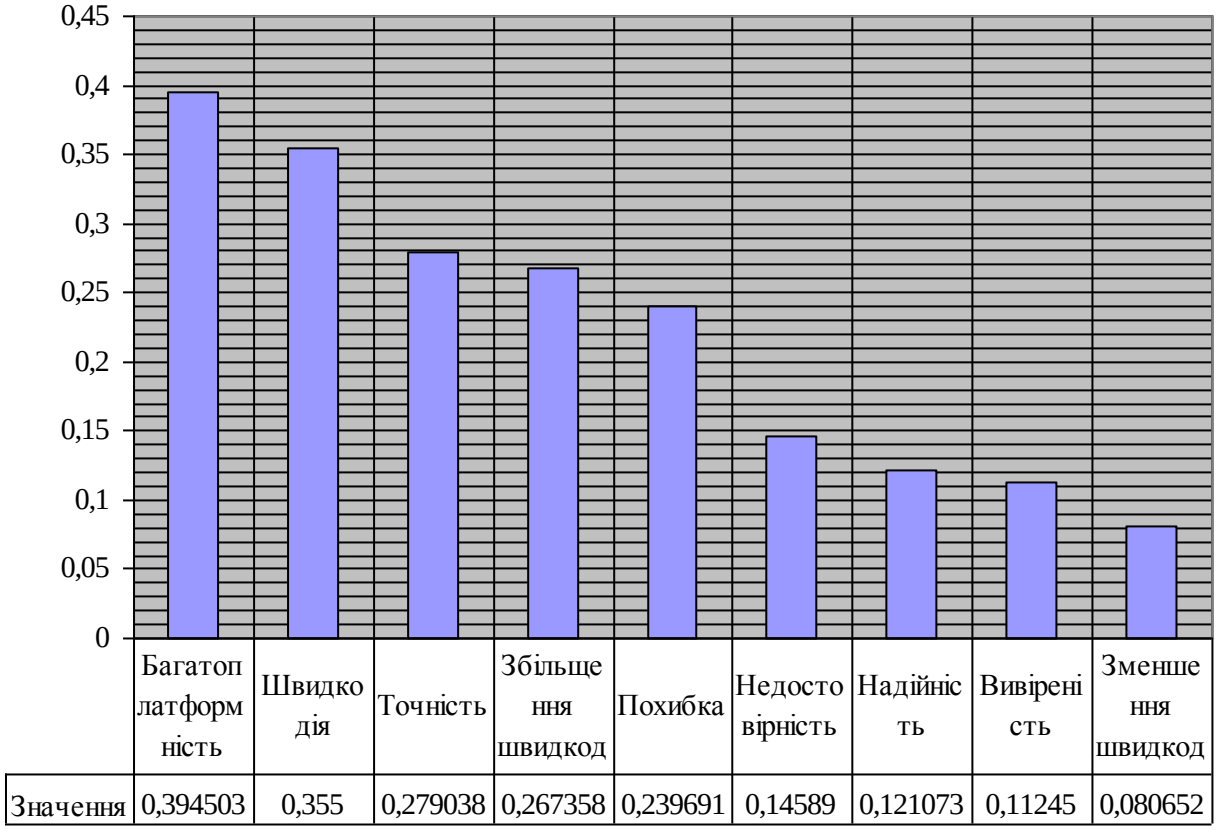
Таблиця 1.9 – Матриця попарних порівнянь критичних властивостей

Критичні властивості	Швидкодія програмної реалізації	Багатоплатформність	Власний вектор	Вектор пріоритетів
Швидкодія програмної реалізації	1	$1/2$	1,22	0,86
Багатоплатформність	2	1	1.41	1,15

Таблиця 1.10 – Матриця попарних порівнянь небажаних властивостей

Небажані властивості	Отримання недостовірних результатів	Накопичення похибки розрахунків	Зменшення швидкодії програмної реалізації моделі	Власний вектор	Вектор пріоритетів
Отримання недостовірних результатів	1	2	$1/3$	0,928	0,199
Накопичення похибки розрахунків	$1/2$	1	8	3,420	0,733
Зменшення швидкодії програмної реалізації моделі	3	$1/8$	1	0,315	0,068

Упевнившись, що табличні дані не суперечать один одному (ІУ $= 0,081$, ВУ $= 0,042$), знайдемо значення вектора глобальних пріоритетів (рис. 1.11).



	Багатоплатформність	Швидкодія	Точність	Збільшення швидкод	Похибка	Недостовірність	Надійність	Вивіреність	Зменшення швидкод
Значення	0,394503	0,355	0,279038	0,267358	0,239691	0,14589	0,121073	0,11245	0,080652

Рисунок 1.11 – Діаграма глобальних пріоритетів

Як бачимо з діаграми 1.11, найбільший вплив мають перші три властивості: проблема використання додатку на різних платформах, швидкодія додатку, та точність отриманих результатів. Так як найвпливовіші властивості припадають на критичні властивості, то гарною практикою буде направити більшу частину ресурсів на покращення цих показників. Також доцільним буде мінімізувати вплив небажаної складової, тому певна частина ресурсів буде виділена і для покращення цих показників.

1.2.3 Модель вирішення проблеми

Розглянемо проблему якісного підвищення показників розрахунків при моделюванні течії в’язкої нестисливої рідини. Ієрархічна структура, що описує модель вирішення проблеми складається із наступних рівнів:

– нульового – проблема, яку необхідно вирішити;

– першого – факторів, які найбільш впливають на проблему;

– другого – акторів, діяльність яких найбільш впливова на проблему;

– третього – сценаріїв вирішення поставленої проблеми.

На основі даних глобального вектора пріоритету незадоволень, у рамках обраного методу були виділені наступні фактори впливу:

– час обробки даних;

– порядок точності;

– багатоплатформність.

Суб’єктами, від чиїх дій залежить результат вирішення проблеми, є:

– дослідник;

– керівник проекту;

У якості сценаріїв вирішення проблеми висунуті наступні альтернативи:

– математичний – алгоритмічна зміна існуючого та розробка нового методу моделювання;

– програмний – заміна технічного приладдя на більш вдосконалене та перехід на спеціалізоване ПО;

– науковий – підвищення кваліфікації дослідників та заохочення нових спеціалістів наукової галузі у досліджені методів моделювання.

На рисунку 1.12 зображена побудована ієрархічна структура, що повністю описує дану модель вирішення поставленої проблеми.

Оскільки більшість показників вже були знайдені, то на основі цих даних зробимо висновок, що найбільш прийнятним рішенням буде «математичний» сценарій, тобто вдосконалення вже існуючого методу, а саме гідродинаміки згладжених частинок.

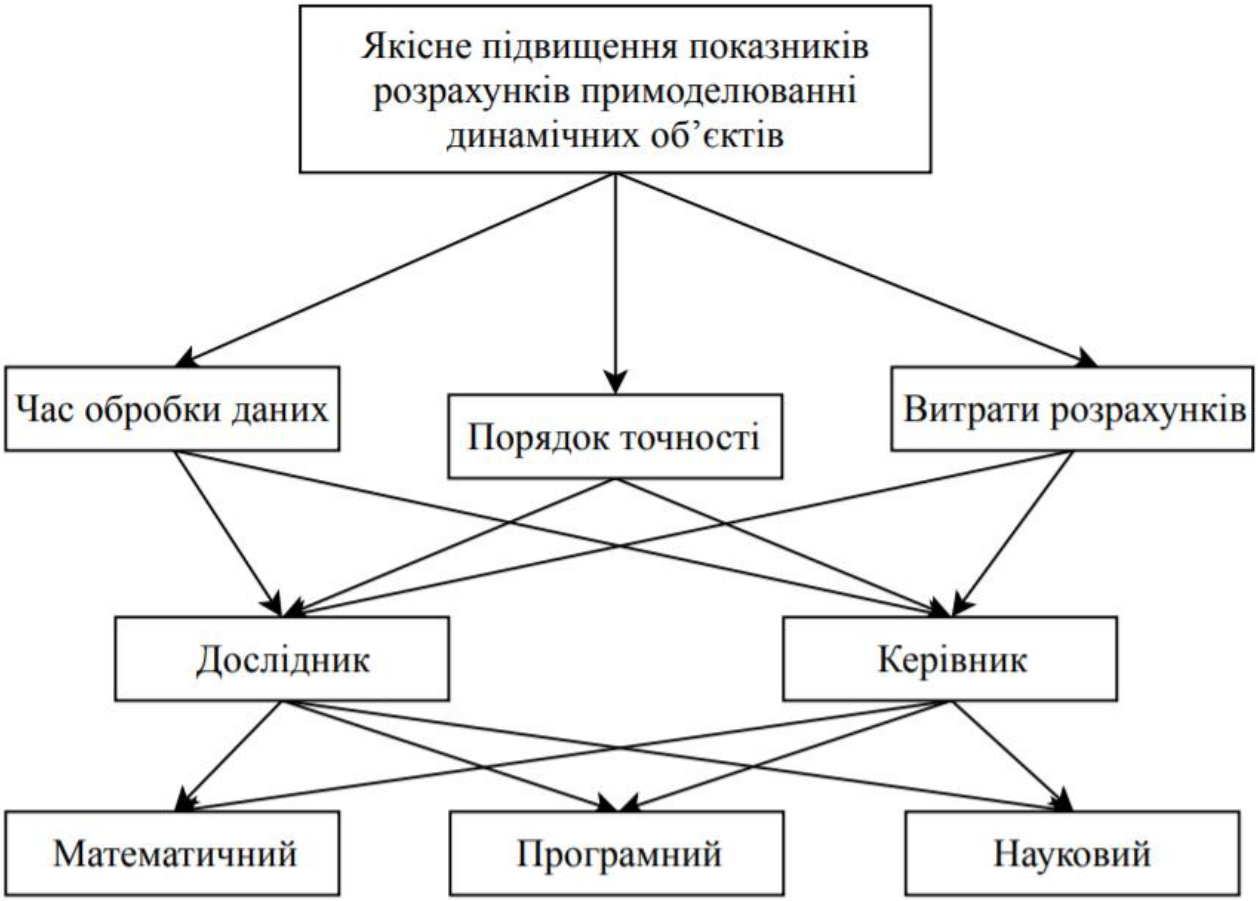


Рисунок 1.12 – Ієрархічна структура вирішення проблеми

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна поставка задачі

Описувати модель системи будемо системою рівнянь Нав’є-Стокса. Рівняння ґрунтується на припущенні, що рідина є одною суцільною речовиною. Більш формально, воно поєднує у собі рівняння руху у суцільному середовищі та рівняння нерозривності та залежить від зовнішніх сил, а також від нестисливості, в’язкості, сили тиску та густини.

Нестисливість показує, що густина не залежить від тиску. Представив рідину у вигляді набору частинок (підхід Лагранжа) спростимо рівняння Нав’є-Стокса. Умовою підходу Лагранжа є те що на всьому часовому проміжку моде-

лювання кількість частинок залишається не змінною, як і їх маса, єдиним виключенням можуть бути зовнішні джерела. Таким чином, рівнянням нерозривності можна знехтувати. Підхід Лагранжа спрощує рівняння руху, через те що після перетворень властивості середовища описуються локальними властивостями частинок і залежить тільки від часу, похідні від швидкості та прискорення спрощуються. В'язкість описується законом внутрішнього тертя Ньютона.

Зовнішні сили можна описати за допомогою другого закону Ньютона.

Внутрішні сили залежать від взаємного розташування частинок. Для того щоб встановити зв'язок між частинками потрібно ввести деяку функцію згладжування, яка залежить від координат та відстані між частинками. Таким чином внутрішні сили будуть залежати лише від функції згладжування.

1.3.2 Формальна постановка задачі

У загальному випадку векторна форма рівняння Нав'є-Стокса, складена з двух рівнянь: рівняння руху у суцільному середовищі, рівняння нерозривності, та записується у наступному вигляді [7]

$$\rho\left(\frac{\partial}{\partial t}+\mathbf{u}\cdot\nabla\right)\mathbf{u}=-\nabla p+\mu\nabla\cdot\left(\nabla\mathbf{u}\right)+\mathbf{f},$$

$$\nabla\cdot\mathbf{u}=0,$$

де $\mathbf{u}:\Omega\times[0,t]\to\mathbb{R}^3$ – швидкість;

$\rho=\text{const}$ – густина рідини;

$p:\Omega\times[0,t]\to\mathbb{R}$ – гідродинамічний тиск рідини;

$\mathbf{f}$ – вектор зовнішньої сили;

μ – динамічна в'язкість рідини;

∇ – оператор набла.

Загальна форма оператора набла (векторний диференціальний оператор, компоненти якого є частинними похідними) в тривимірному просторі має вигляд

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right).$$

Застосувавши підхід Лагранжа спростимо основне рівняння знехтувавши рівнянням нерозривності, після чого основне рівняння перепишеться в наступну форму:

$$\rho \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{F}. \qquad (1.1)$$

Права частина рівняння (1.1) складається із зовнішніх та внутрішніх сил, які можуть бути записані у вигляді суми $\mathbf{F} = \mathbf{f}^{зовн.} + \mathbf{f}^{внут.}$. На основі цього стає можливим розрахунок прискорення *i*-ої частинки, що набуває наступного вигляду

$$\mathbf{a}_i = \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{\mathbf{F}_i}{\rho_i}, \qquad (1.2)$$

де $\mathbf{a}_i$ – вектор прискорення i-ої частинки;

$\mathbf{u}_i$ – вектор швидкості i-ої частинки;

$\mathbf{F}_i$ – сума зовнішніх та внутрішніх сил i-ої частинки;

ρ_i – густина i-ої частинки.

1.4 Постановка задач дослідження

Опираючись на проведений системний аналіз та постановку задачі, сформулюємо задачі магістерської роботи:

– на основі методу гідродинаміки згладжених частинок та *R*-функцій оптимізувати алгоритм, що моделює течію в’язкої нестисливої рідини, яка описується лінійним диференціальним рівнянням (1.1), та взаємодію течії з довільним геометричним об’єктом, побудованим за допомогою алгоритму крокуючих кубиків у тривимірному просторі, шляхом виконання чисельних обчислень на графічному процессорі;

– оптимізувати алгоритм гідродинаміки згладжених частинок мовою програмування C++ та за допомогою бібліотеки CUDA виконати обчислення на графічному процессорі та візілізувати результати у тривимірному просторі за допомогою графічної бібліотеки OpenGL;

– провести обчислювальні експерименти з різними початковими положеннями часток та з різними довільними геометричними об’єктами всередині суцільного середовища;

– покращити кросплатформену програму для ОС Windows, MacOS та Linux.

2 ВИБІР ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Метод *R*-функцій та обернена задача аналітичної геометрії в 2D

Множина *R*-функцій є множина, що перетинається із множиною елементарних функцій [8]. Завдяки цьому існує можливість виконувати над *R*-функціями диференціювання та інші розрахункові операції. Завдячуючи одночасному розгляду функцій дискретного та неперервного аналізу, було помічено існування серед функцій неперервних аргументів *R*-функцій, що по своїм властивостям нагадують функції алгебри логіки. Таким чином, кожній *R*-функцій поставлена у відповідність булева функція, що надало можливість надалі використовувати методи алгебри логіки у класичному неперервному аналізу для опису складних геометричних об'єктів.

Сформулюємо обернену задачу аналітичної геометрії та розглянемо основні поняття теорії *R*-функцій [8, 9]. Нехай у $\mathbb{R}^2$ задано геометричний об'єкт Ω з кусково-гладкою границею $\partial\Omega$ і вимагається побудувати функцію $\omega(x, y)$ додатну всередині Ω, від'ємну по за межами Ω і нульову на $\partial\Omega$. Рівняння $\omega(x, y) = 0$ буде у неявній формі визначати геометричне місце точок, що представляють границю $\partial\Omega$ області Ω. Надалі будемо також використовувати термін локус, під назвою якого слід розуміти множину точок $\mathbb{R}^2$, для якої за допомогою деякої системи елементарних функцій та констант $a \in \mathbb{R}$ шляхом утворення їх суперпозицій можна написати рівняння виду $\omega(x, y) = 0$. Причому функція $\omega(x, y)$ буде являтися елементарною функцією та мати вид єдиного аналітичного виразу. Функція $\omega(x, y)$ може бути достатньо просто побудована за допомогою методу *R*-функцій для областей довільної форми.

Визначення 1.1. *R*-функцією, що відповідає розбиттю числової осі на інтервали $-\infty,\ 0$ і $0,\ +\infty$, називається така функція, знак якої цілком визначається знаками її аргументів.

Існує також і альтернативне визначення.

Визначення 1.2. Функція $y = f(x_1,...,x_n)$, яка визначена усюди в $\mathbb{R}^n$, називається *R*-функцією, якщо існує така булева функція F, що виконується рівність

$$S\left[f(x_1,...,x_n)\right] = F\left[S(x_1),..., S(x_n)\right], \tag{2.1}$$

де двозначний предикат $S(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$

Визначення 1.3. Функція F із рівняння (2.1) називається супроводжувальною для *R*-функції $y = f(x_1,...,x_n)$.

Кожній *R*-функції відповідає єдина супроводжувальна функція, зворотне невірно.

Визначення 1.4. Сукупність усіх *R*-функцій, що мають одну і ту саму супроводжувальну булеву функцію, назвемо гілкою множини *R*-функцій. Усього існує 2^{2^n} гілок множини *R*-функцій n змінних.

Визначення 1.5. Система H *R*-функцій називається достатньо повною, якщо перетин множини суперпозицій H – з кожною гілкою множини *R*-функцій не являється пустим.

Розглянемо основні достатньо повні системи *R*-функцій, що відповідають системи супроводжувальних функцій

$$H_2 = \{0, \overline{X}, X_1 \wedge X_2, X_1 \vee X_2, X_1 \sim X_2, X_1 \Rightarrow X_2, X_1 \mid X_2\}$$

Найчастіше використовується система $\Re_\alpha$:

$$x \wedge_\alpha y \equiv (1+\alpha)^{-1}\left(x + y - \sqrt{x^2 + y^2 - 2\alpha xy}\right),$$

$$x \vee_\alpha y \equiv (1+\alpha)^{-1}\left(x + y + \sqrt{x^2 + y^2 - 2\alpha xy}\right),$$

$$\overline{x} \equiv -x.$$

Зауважимо: $-1 < \alpha(x, y) \le 1,\ \alpha(x, y) \equiv \alpha(y, x) \equiv \alpha(-x, y) \equiv \alpha(x, -y)$.

При $\alpha \equiv 0$ отримаємо систему $\Re_0$, яку часто використовують на практиці

$$x \wedge_0 y \equiv x + y - \sqrt{x^2 + y^2},$$

$$x \vee_0 y \equiv x + y + \sqrt{x^2 + y^2},$$

$$\overline{x} \equiv -x.$$

Нехай область Ω може бути сконструйована з опорних областей $\Sigma_1,\ \Sigma_2,\ ...,\ \Sigma_m$ за логічними правилами, що визначаються булевою функцією F, за допомогою операції кон'юнкції, диз'юнкції та заперечення: $\Omega = F(\Sigma_1,\ \Sigma_2,\ ...,\ \Sigma_m)$.

Вважаємо, що кожна опорна область $\Sigma_i,\ i = 1,\ 2,\ ...,\ m$, може бути задана у вигляді $\Sigma_i = \{\omega_i(x, y) \ge 0\}$, де функція $\omega_i(x, y)$ являється елементарною. Тоді, замінюючи у виразі для Ω символи кон'юнкції, диз'юнкції та заперечення на символи *R*-операцій $\wedge_\alpha,\ \vee_\alpha,\ -$, а Ω та $\Sigma_i,\ i = 1,\ 2,\ ...,\ m$ на $\omega(x, y)$ і $\omega_i(x, y)$, $i = 1,\ 2,\ ...,\ m$, відповідно, отримаємо аналітичний вираз, що визнає в елементарних функціях необхідне рівняння границі $\omega(x, y) = 0$. При цьому для внутрішніх точок області $\omega(x, y) > 0$, а для зовнішніх $\omega(x, y) < 0$.

R-функції знайшли своє застосування у різних напрямках, таких як оптимальне розміщення геометричних об'єктів, математичне програмування, конструктивна теорія функцій та особливо крайові задачі математичної фізики [1]. Проте і цим не завершується перелік сфер використання методу.

Потужним поштовхом у теорії стійкого руху було застосування *R*-функції у досліджені області стійкості невстановленого руху, надавши онов-

лені результати числової реалізації класичних критеріїв стійкості, побудови функції Ляпунова, в задачі стійкості аналітичних рухів тощо. Перспективними областями є також розпізнання образів, хімічні технології, медична діагностика та інші.

2.2 Розв'язок оберненої задачі аналітичної геометрії в 3D

З методами побудови рівнянь ГО (Геометричний об'єкт) на основі методу R-функцій добре поєднуються класичні способи побудови рівнянь поверхонь тіл обертання, призматичних и конусових тіл, скручених циліндрів і змійовиків некласичного поперечного перерізу.

Розглянемо, перш за все, методи побудови нормалізованих рівнянь поверхонь тіл обертання. Нехай $\omega_0(x,y)=0$ – нормалізоване рівняння межі області $\Omega_0=(\omega_0(x,y)\geq 0)$. Тоді рівняння виду

$$\omega(x,y,z)\equiv\omega_0(x,\sqrt{y^2+z^2})=0 \qquad (2.2)$$

являється рівнянням ГО Ω, для якого вісь Ox являється віссю симетрії.

Цей добре відомий спосіб аналітичної геометрії має дві особливості – позитивну і негативну.

Нехай $\omega_0(x,y)=0$ є рівнянням ГО Ω_0. Тоді $\omega_0(x,-y)=0$ є рівнянням дзеркального відображення Ω_0 відносно осі Ox. Очевидно, що обидва рівняння належать до одного й того ж диференціального класу (наприклад, $C^m(R^2)$) і одночасно нормалізовані чи не нормалізовані до одного чи іншого порядку. Нехай на початку $\omega_0(x,y)=0$ є така межа області $\Omega_0=(\omega_0(x,y)\geq 0)$. Тоді задана нерівність $\omega_0(x,-y)=0$ визначає область іншої Ω_0', яка є дзеркальним відображенням області Ω_0 відносно осі Ox. Нерівність

$\omega_0^*(x,y) \equiv \omega_0(x,y) \vee_0 \omega_0(x,-y) \geq 0$ визначає з'єднання областей Ω_0 і Ω_0'. Нехай область Ω_0 не пересікається с віссю *Ox*. Тоді не важко визначити, що функція $\omega_0^*(x,y)$ належить одному й тому ж диференціальному класу, що і функція $\omega_0(x,y)$. Застосувавши к $\omega_0^*(x,y)=0$ формулу (1.3), рівняння розглянутої поверхні обертання отримаємо в вигляді

$$\omega(x,y,z) \equiv \omega_0(x,\sqrt{y^2+z^2}) \vee_0 \omega_0(x,-\sqrt{y^2+z^2}) = 0. \qquad (2.3)$$

Звідси маємо

$$\frac{\partial \omega(x,y,z)}{\partial y} = \frac{\partial \omega_0^*(x,\sqrt{y^2+z^2})}{\partial y} \frac{y}{\sqrt{y^2+z^2}}\Bigg|_{y=z=0} = 0.$$

Аналогічно маємо, що і $\frac{\partial \omega(x,0,0)}{\partial z} = 0$. Таким чином, відмінно від рівняння (2.2), рівняння (2.3) не має на осі обертання розривів похідних. В той самий час в силу відомих властивостей достатньо повної системи $\{R_0\}$ рівняння (2.3) нормалізовано до першого порядку, якщо нормалізоване рівняння $\omega_0(x,y)=0$.

Один з корисних підходів к побудові ГО в 3D з 2D полягає в введенні функціональних параметрів, а саме, якщо в області *xOy* межа ГО описується рівнянням

$$\omega_0(x,y,c_1,...,c_m) = 0, \qquad (2.4)$$

де c_i – геометричні параметри, визначаючі форму і розміри ГО Ω_0 і його елементів, то, вводячи функції $c_i(z)(i=1,...,m)$, при умові $c_i(0)=c_i$, отримаємо рівняння поверхні виду

$$\omega(x, y, c_1(z), ..., c_m(z)) = 0,$$

якому в перетинах $z = h = const$ будуть відповідати ГО з сімейства (2.4).

Наприклад, якщо відомо, що при $z = 0$ межа ГО має вид прямокутника зі сторонами $x = \pm a,\ y = \pm b$ і описується рівнянням $(a^2 - x^2) \wedge_0 (b^2 - y^2) = 0$, а при $z = H$ межа ГО – також прямокутник зі сторонами $x = \pm c,\ y = \pm d,$ і описується рівнянням $(c^2 - x^2) \wedge_0 (d^2 - y^2) = 0,$ то, дотримуючись вищевикладеного, можна записати рівняння бокової поверхні с лінійною залежністю по $z,$ з'єднуючої прямокутники, в вигляді

$$\partial\Omega = \left(\left[a\left(1 - \frac{z}{H}\right) + c\frac{z}{H} \right]^2 - x^2 \right) \vee_0 \left(\left[b\left(1 - \frac{z}{H}\right) + d\frac{z}{H} \right]^2 - y^2 \right) = 0).$$

Розглянемо декілька прикладів.

Приклад 1. Куб, який визначається вершинами $O(0,0,0)$, $A(0,0,3)$, $B(0,3,0)$, $C(0,3,3)$, $D(3,0,0)$, $E(3,0,3)$, $F(3,3,0)$, $G(3,3,3)$. Куб можна уявити як

$$\Omega = \Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3,$$

де $\Sigma_1 = (z(3 - z) \geq 0)$ – область обмежена площинами $z = 0$ та $z = 3;$

$\Sigma_2 = (x(3 - x) \geq 0)$ – площинами $x = 0$ та $x = 3;$

$\Sigma_3 = (y(3 - y) \geq 0)$ – площинами $y = 0$ та $y = 3.$

Тоді рівняння $\partial\Omega$ має вигляд

$$\partial\Omega = \omega(x, y, z) = [z(3 - z) \wedge_0 x(3 - x)] \wedge_0 y(3 - y) = 0$$

Приклад 2. Пішака (рис. 2.1) можна описати як

$$\Omega = (\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3) \vee (\Sigma_4 \vee \Sigma_5).$$

де $\Sigma_1 = (\omega_1 \geq 0)$, $\omega_1 = 0.25 - (x-1.5)^2 - (y-1.5)^2$ – циліндр;

$\Sigma_2 = (\omega_2 \geq 0)$, $\omega_2 = z(1-z)$ – область між $z = 0$ та $z = 1$;

$\Sigma_3 = (\omega_3 \geq 0)$, $\omega_3 = -20((x-1.5)^2 + (y-1.5)^2) + 1 + 10(z-0.75)^2$ – гіперболоїд;

$\Sigma_4 = (\omega_4 \geq 0)$, $\omega_4 = 0.125 - (x-1.5)^2 - (y-1.5)^2 - 20(1-z)^2$ – еліпсоїд;

$\Sigma_5 = (\omega_5 \geq 0)$, $\omega_5 = 0.05 - (x-1.5)^2 - (y-1.5)^2 - (1.25-z)^2$ – сфера.

Рівняння $\partial\Omega$ можна описати як:

$$\omega(x, y, z) = [\omega_1 \wedge_0 \omega_2 \wedge_0 \omega_3] \vee_0 [\omega_4 \vee_0 \omega_5] = 0$$

Приклад 3. Слона (рис. 2.2) можна описати як

$$\Omega = (\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3) \vee (\Sigma_4 \vee \Sigma_5),$$

де $\Sigma_1 = (\omega_1 \geq 0)$, $\omega_1 = 0.25 - (x-1.5)^2 - (y-1.5)^2$ – циліндр;

$\Sigma_2 = (\omega_2 \geq 0)$, $\omega_2 = z(1.25-z)$ – область між $z = 0$ та $z = 1.25$, $\Sigma_3 = (\omega_3 \geq 0)$;

$\omega_3 = -20((x-1.5)^2 + (y-1.5)^2) + 1 + 10(z-0.85)^2$ – гіперболоїд, $\Sigma_4 = (\omega_4 \geq 0)$;

$\omega_4 = 0.2 - (x-1.5)^2 - (y-1.5)^2 - 20(1.25-z)^2$ – еліпсоїд, $\Sigma_5 = (\omega_5 \geq 0)$;

$\omega_5 = 0.2 - 5(x-1.5)^2 - 4(y-1.5)^2 - (1.4-z)^2$ – еліпсоїд.

Рівняння $\partial\Omega$ можна описати як:

$$\omega(x, y, z) = [\omega_1 \wedge_0 \omega_2 \wedge_0 \omega_3] \vee_0 [\omega_4 \vee_0 \omega_5] = 0$$

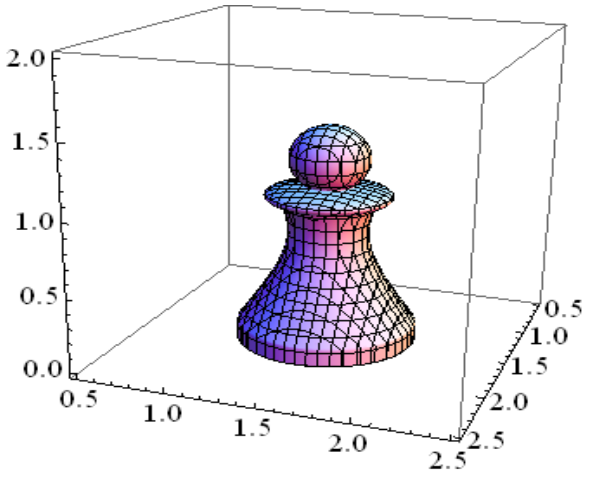


Рисунок 2.1 – Пішак

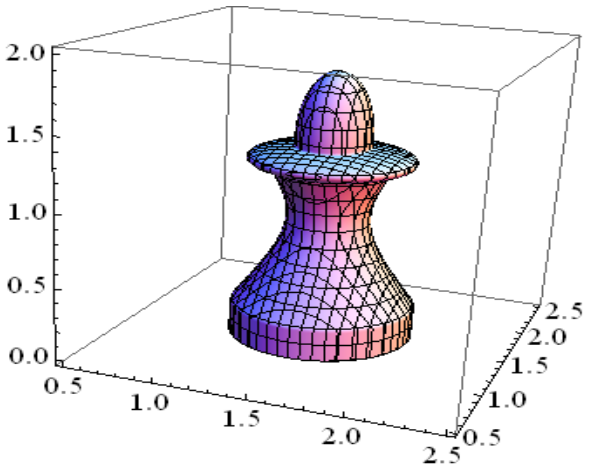


Рисунок 2.2 – Слон

2.3 Алгоритм крокуючих кубиків

Припустимо, що у нас є функція $F(x,y,z)=0$, яку можна дискретизувати на всьому просторі. Функція – це відмінний спосіб опису довільної фігури, але вона не допомагає нам побудувати її. В цьому нам допоможе алгоритм крокуючих кубиків. Алгоритм розбиває простір на дискретну кількість кубів та визначає як поверхня перетинає ці куби. Поверхня обробляється у послідовному куб-за-кубом чином. Таким чином, метод спочатку обробляє m кубів першого ряду даних в послідовному порядку $C_1, C_2, ..., C_m$. Під час цієї обробки кожна верши-

на куба V_i, в якій значення більше або дорівнює нулю помічається, а всі інші вершини залишаються без позначки. Оскільки кожна з восьми вершин куба може бути позначена або непозначена, то існує 256 (2^8) можливих сценаріїв маркування для куба. Кожен сценарій розмітки куба кодує шаблон перетину куба з поверхнею.

Алгоритм розглядається як рефлексивний і рефлекторно симетричний, що призводить до 15 унікальних сценаріїв маркування. Куби C і $\hat{C}$ є рефлекторно симетричними, якщо кожна вершина в положенні V_i на C має протилежну мітку як вершини в тому ж положенні V_i в $\hat{C}$. Два рефлекторно симетричних показані в центрі і праворуч (тобто, куби A і A_F) на рисунку (рис. 2.3).

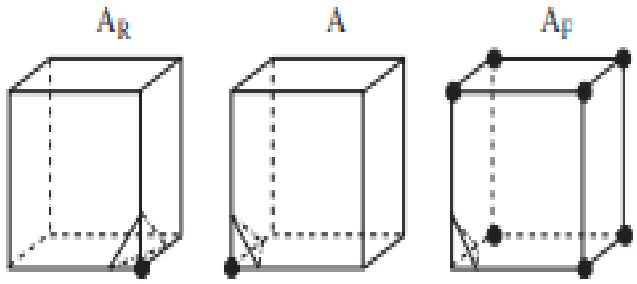


Рисунок 2.3 – Ілюстрація рефлективної (A з A_F)
і обертальної (A з A_P) симетрії

Куби, які є рефлекторно симетричними, мають однакову схему перетину куба і поверхні. Два куби C і $\hat{C}$ ротаційно симетричні, якщо існує деяка серія обертів R, яка при застосуванні до C перетворює C в нову орієнтацію, в якій розмітка в кожному перетвореному положенні вершини V_i є ідентичною міткою в тому ж положенні V_i у $\hat{C}$. Куби A і A_P в на рисунку (рис. 2.3) обертально симетричні. 15 унікальних сценаріїв перетину куба-ізоповерхні, що виникають при розгляді обох цих симетрій, представлені на рисунку 2.4

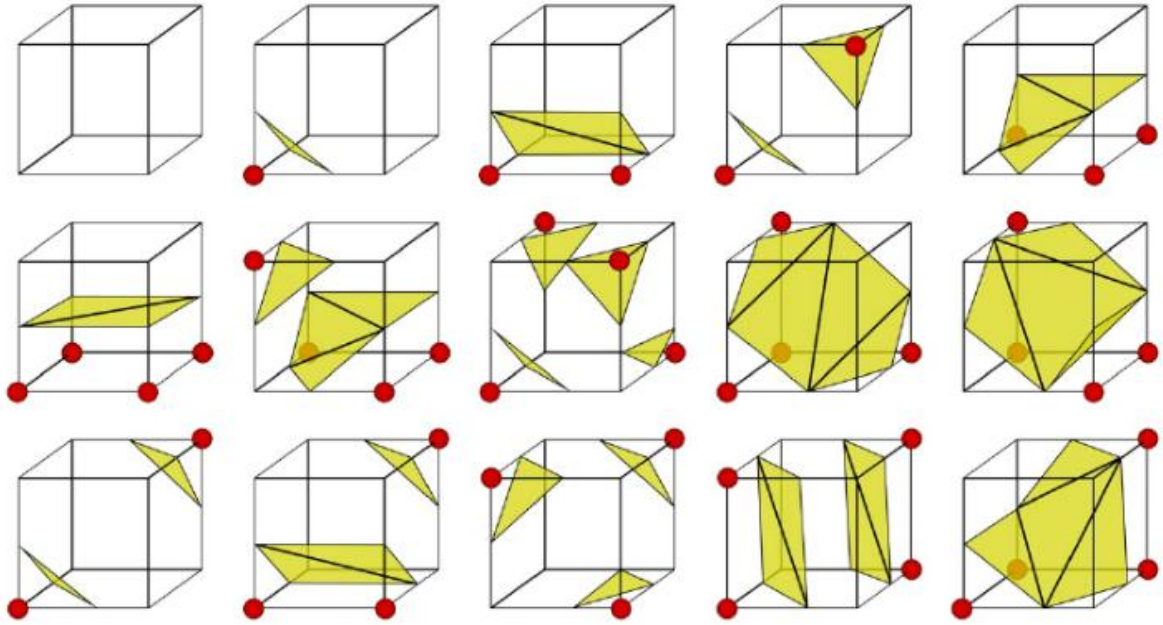


Рисунок 2.4 – 15 основних сценаріїв перетину

Місця перетину поверхневого краю можна оцінити з точністю до вершини, використовуючи методику інтерполяції. В методі використовується лінійна інтерполяція для оцінки точки перетину для кожного пересіченого краю. Якщо ребро одиничної довжини E має кінцеві точки V_s і V_e, скалярними значеннями яких є L_s і L_e відповідно, тоді задана площина α, в місті перетину $I = (I_x, I_y, I_z)$ має компоненти форми:

$$I_{\{x,y,z\}} = V_{S_{\{x,y,z\}}} + \rho(V_{e_{\{x,y,z\}}} - V_{s_{\{x,y,z\}}}),$$

де $\rho = \dfrac{\alpha - L_s}{L_e - L_s}$.

Останнім кроком методу є генерування трикутних граней, які представляють частину поверхні, що перетинає кожен куб. Точки перетину визначають вершини трикутників, а збірка трикутних граней по всіх кубах утворює трикутну сітку (або сітку), що визначає поверхню.

2.4 Гідродинаміка згладжених частинок

Гідродинаміка згладжених частинок (Smoothed Particle Hydrodynamics – SPH) являє собою метод апроксимації динаміки руху рідини через використання частинок, які можуть бути представлені як інтерполяційні точки. Основна ідея SPH – розгляд гідродинамічних процесів за допомогою дискретної кількості частинок.

В основі методу лежить використання локальних інтерполяцій на оточуючих дискретних частинках для побудови неперервного поля апроксимацій яке може буде використано для дискретизації рівнянь поля. Інтерполяція виконується за допомогою функції ядра. Для даної задачі раціональним буде використання окремого випадку застосування ядерного інтерполянта сумування для знаходження густини, через яку буде виражена решта гідродинамічних рівнянь. В основі методу лежить інтерполяція за допомогою ***функції ядра***:

$$\int_{\Omega} W(\mathbf{r},h)\,\mathrm{d}\mathbf{r}=1,\;\; W(\mathbf{r},h)\geq 0,$$

$$W(\mathbf{r},h)=W(-\mathbf{r},h),$$

$$W(\mathbf{r},h)=0 \Leftrightarrow \|\mathbf{r}\|>h,$$

де $\mathbf{r}$ – довільна точка у Ω,

h – радіус згладжування, параметр гладкості інтерполяції.

Для будь-якої функції $F(\mathbf{r})$ може бути знайдений її згладжений інтерполянт $F_s(\mathbf{r})$ через згортання ядром $W(\mathbf{r},h)$:

$$F_s(\mathbf{r})=\int F(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)\,\mathrm{d}\mathbf{r}'.$$

Зазвичай за функцію ядра беруть Гаусове ядро, бо воно має незначний вплив на віддалені частинки. Проте для збільшення продуктивності змінимо

ядро на сплайн, значення якого за межами двох радіусів згладжування обнулюється. Дане спрощення значно зменшує кількість розрахунків, через що значення будь-якої фізичної величини визначається відповідними значеннями сусідніх частинок, які потрапили у охоплювану радіусом область.

Для використання сплайну у подальших розрахунках вимагаємо симетричність та достатню гладкість, отримавши таким чином двічі неперервно-диференційоване ядро. Відповідний сплайн шостого ступеня приймає вигляд:

$$W(\mathbf{r},h)=\frac{315}{64\pi h^9}\begin{cases}\left(h^2-\|\mathbf{r}\|^2\right)^3, & 0\le\|\mathbf{r}\|\le h,\\ 0, & \|\mathbf{r}\|>h.\end{cases} \tag{2.6}$$

Припустимо, що нам відома множина усіх точок $\mathbf{r}_i$. Маючи загальну масу та густину частинок, легко знайти загальний об’єм кінцевого елемента. У випадку, коли точки являються достатньо густою вибіркою об’єму ядра, апроксимуємо вираз (2.6), замінивши інтеграл сумою

$$F_s(\mathbf{r})=\sum_j \frac{\mathrm{m}_j}{\rho_j}F(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j,h), \tag{2.7}$$

де m_j – маса частинки;

ρ_j – густина частинки;

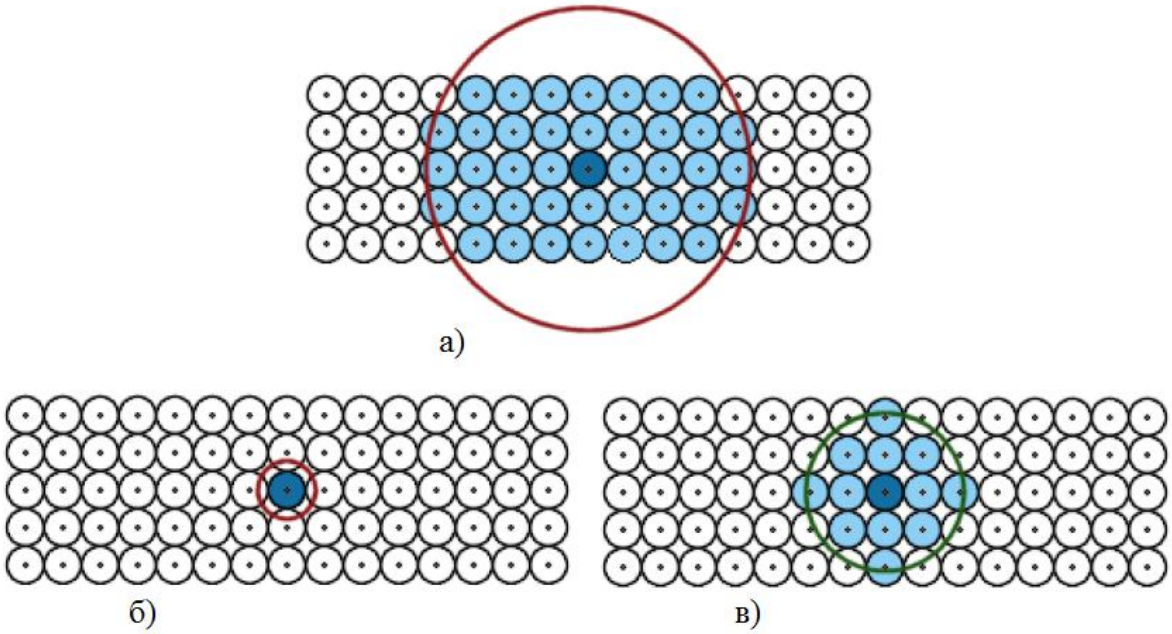
$\Delta\mathbf{r}_j\approx\frac{\mathrm{m}_j}{\rho_j}$ – кінцевий об’єм елемента.

Важливим є те, що наближення $F_s(\mathbf{r})$ визначене усюди та диференційоване завдячуючи диференційованості ядра, попри значну похибку інтерполяції похідної. У випадку, якщо $F(\mathbf{r})=\rho(\mathbf{r})$, отримаємо наближення густини

$$\rho_s(\mathbf{r}) \approx \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h), \tag{2.8}$$

що залежить лише від маси та координати частинки.

В загальному випадку довжина згладжування може залежати від простору, $h = h(\mathbf{r})$, на основі чого виникає можливість розрахунку варіацій у густині вибірки.



а) великий радіус; б) малий радіус; в) добре підібраний

Рисунок 2.5 – Підбір відповідного значення радіусу згладжування

Таким чином, оскільки єдиною вимогою є лише значення густини частинки, а загальна маса залишається постійною, то однозначна рівність між інтегралом загального вигляду по об'єму $\rho_s(\mathbf{r})$ та загальною масою є неважливою.

Тому використаємо підхід названий «збиранням», де у якості ядра (2.8) виступає $W(\mathbf{r}_i - \mathbf{r}_j, h(\mathbf{r}_i))$, і потребує лише значення h_i для визначення густини i-ої частинки. Через що, використовуючи підхід «збирання» значно спрощується рівняння (2.8) і його форма запису набуває вигляду

$$\rho_i = \sum_{j=1}^{N} \mathrm{m}_j W\left(\mathbf{r}_i - \mathbf{r}_j, h_i\right), \tag{2.9}$$

де N – кількість частинок у межах кола радіусу $2h$ навколо точки $\mathbf{r}_i$.

Застосувавши підхід інтерполяції ядра, виразимо й решту гідродинамічних показників.

Розглянемо внутрішні сили, які налічують у собі тиск та в'язкість. Для вираження тиску скористаємося законом ідеального газу

$$pV = nRT,$$

де $V = \dfrac{1}{p}$ – об'єм на одиницю маси;

n – кількість частинок на одну моль;

R – універсальна газова константа;

T – температура.

Для ізотермічної рідини із постійною масою права частина рівняння замінюється константою Больцмана

$$\begin{aligned} pV &= \kappa, \\ p\frac{1}{\rho} &= \kappa, \\ p &= \kappa\rho. \end{aligned} \tag{2.10}$$

Якщо тиск для кожної частинки відомий, то сила тиску на i-у частинку, представлена через інтерполяцію ядра, набуває вигляду:

$$\mathbf{f}_i^{тиск} = -\sum_{j \neq i} p_i \frac{\mathrm{m}_j}{\rho_i} \nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right).$$

Проте отримана сила тиску не є симетричною, що може бути перевірено при взаємодії лише двох частинок. Задля вирішення цієї проблеми можна скористатися властивостями SPH

$$\mathbf{f}_i^{тиск} = -\rho_i \sum_{j \neq i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \mathrm{m}_j \nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right), \tag{2.11}$$

або використовуючи загальні фізичні властивості

$$\mathbf{f}_i^{тиск} = -\sum_{j \neq i} \frac{p_i + p_j}{2} \frac{\mathrm{m}_j}{\rho_j} \nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right). \tag{2.12}$$

Таким чином, отримаємо силу тиску, що зберігає імпульс та кутовий момент, тим самим задовольняючи третій закон Ньютона [10].

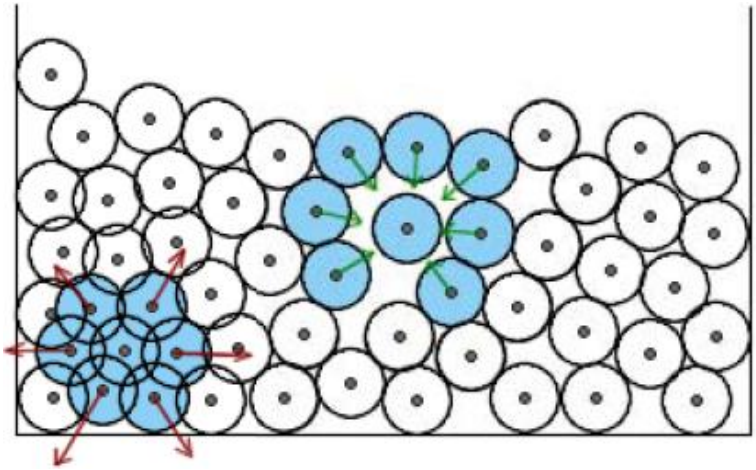


Рисунок 2.6 – Поведінка сили тиску при великій та малій густині

Використовуючи загальну формулу інтерполяційного ядра (2.5) та отримане рівняння для сили тиску, отримаємо відповідне інтерполяційне ядро, що описує взаємодію між частинками, та градієнт

$$W_{тиск}(\mathbf{r},h)=\frac{15}{\pi h^6}\begin{cases}\left(h-\|\mathbf{r}\|\right)^3, & 0\le\|\mathbf{r}\|\le h,\\ 0, & \|\mathbf{r}\|>h,\end{cases}$$

$$\nabla W_{тиск}(\mathbf{r},h)=-\frac{45}{\pi h^6}\frac{\mathbf{r}}{\|\mathbf{r}\|}\left(h-\|\mathbf{r}\|\right)^2,$$

$$\lim_{\mathbf{r}\to-0}\nabla W_{тиск}(\mathbf{r},h)=\frac{45}{\pi h^6},\quad \lim_{\mathbf{r}\to+0}\nabla W_{тиск}(\mathbf{r},h)=-\frac{45}{\pi h^6}.$$

Загальна формула для сили в’язкості у термінах SPH має вигляд

$$\mathbf{f}_i^{в'язк.}=\mu\sum_{j\neq i}\mathbf{u}_j\frac{\mathrm{m}_j}{\rho_i}\nabla^2W\left(\mathbf{r}_i-\mathbf{r}_j,h\right).$$

Аналогічно силі тиску, сила в’язкості також не симетрична відносно зміни швидкості від частинки до частинки. В протидію цьому було використано наступне

$$\mathbf{f}_i^{в'язк.}=\mu\sum_{j\neq i}\left(\mathbf{u}_j-\mathbf{u}_i\right)\frac{\mathrm{m}_j}{\rho_i}\nabla^2W\left(\mathbf{r}_i-\mathbf{r}_j,h\right),\qquad(2.13)$$

де сила в’язкості залежить лише від різниць швидкостей, а не від їх абсолютних значень. Відповідне інтерполяційне ядро та градієнт набувають вигляду:

$$W_{в'язк.}(\mathbf{r},h)=\frac{15}{2\pi h^3}\begin{cases}-\frac{\|\mathbf{r}\|^3}{2h^3}+\frac{\|\mathbf{r}\|^2}{h^2}+\frac{h}{2\|\mathbf{r}\|}-1, & 0\le\|\mathbf{r}\|\le h,\\ 0, & \|\mathbf{r}\|>h,\end{cases}$$

$$\lim_{\mathbf{r}\to0}W_{в'язк.}(\mathbf{r},h)=\infty,$$

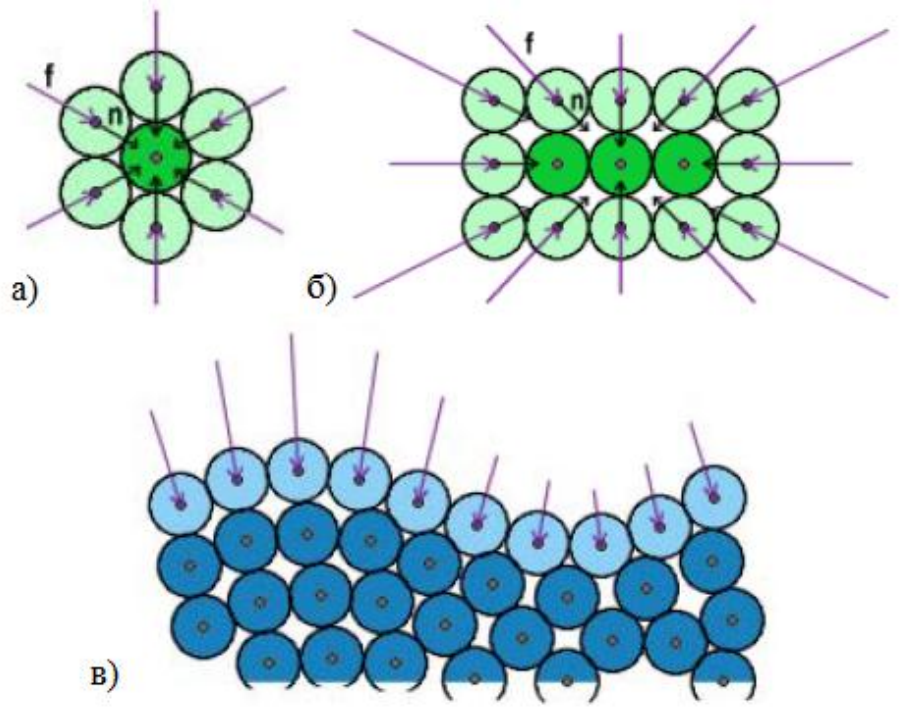
$$\nabla W_{в'язк.}(\mathbf{r},h)=\frac{15}{2\pi h^3}\mathbf{r}\left(-\frac{3\|\mathbf{r}\|}{2h^3}+\frac{2}{h^2}+\frac{h}{2\|\mathbf{r}\|^3}\right),$$

$$\lim_{\mathbf{r}\to-0}\nabla W_{в'язк.}(\mathbf{r},h)=+\infty,\quad \lim_{\mathbf{r}\to+0}\nabla W_{в'язк.}(\mathbf{r},h)=-\infty.$$

Оскільки зовнішні сили складаються лише із сили гравітації та її формулювання тривіально виражається через другий закон Ньютона

$$\mathbf{f}_i^{тяж.} = \rho_i \mathbf{g}, \tag{2.14}$$

то залишається одна сила – сила поверхневого натягу, яка для певної множини частинок є додатковою зовнішньою силою. Вона не є частиною рівняння Нав'є-Стокса, бо являється граничною умовою. У підході Лагранжа межа представляється множиною частинок.



а) сферичної форми; б) капсульної форми; в) кривої хвилі

Рисунок 2.7 – Поводження поверхневого натягу

На молекули рідин впливають сили сусідніх молекул, що знаходяться у ідеальному балансі усередині рідини. На поверхні ж сили розбалансовані – це і спричиняє поверхневий натяг. Сила поверхневого натягу співнапрямлена із поверхневими нормалями відносно рідини, де вони зв'язують її поверхню разом. Сила поверхневого натягу розпрямлює поверхневу кривизну, мінімізуючи площу

поверхні. На рисунку 2.7 зображена типова поведінка сили поверхневого натягу при різних умовах. Модель опису поверхневого натягу описується, як

$$\mathbf{f}_i^{пов.} = -\sigma\nabla^2 c_i \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}, \tag{2.15}$$

де $\mathbf{n}_i$ – напрямлена усередину поверхнева нормаль рідини i-ої частинки;

c_i – згладжена величина кольору частинки;

σ – коефіцієнт тертя.

Згладжена величина кольору частинки є додатковою величиною у SPH і задається у наступному вигляді

$$c_i = \sum_j \frac{\mathrm{m}_j}{\rho_i} W\left(\mathbf{r}_i - \mathbf{r}_j, h\right).$$

Основною властивістю цієї величини є те, що її градієнт є поверхневою нормаллю

$$\mathbf{n}_i = \nabla c_i = \sum_j \frac{\mathrm{m}_j}{\rho_i} \nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right), \tag{2.16}$$

де $\|\mathbf{n}_i\| > 0$ тільки біля або на поверхні рідини. Дивергенція нормалі вимірює Гаусову кривизну поверхні

$$\kappa = -\frac{\nabla \mathbf{n}}{\|\mathbf{n}\|} = -\frac{\nabla^2 c}{\|\mathbf{n}\|},$$

від'ємність якої необхідна для отримання позитивного об'єму випуклої кривизни. Поверхнева сила зціплення є силою, що діє на одиницю площі даної поверхні рідини і розподіляється по навколишніх частинках

$$\mathbf{t} = \sigma\kappa \frac{\mathbf{n}}{\|\mathbf{n}\|}.$$

Оскільки $\|\mathbf{n}_i\|$ стає зменшується із віддаленням i-ої частинки від поверхні, можемо помножити поверхневе зціплення на нормалізовану скалярну величину $\delta_i = \|\mathbf{n}_i\|$. За рахунок цього впевнимося, що сила густини розподілена по усіх можливих частинках. З огляду на це, виведемо наступну рівність

$$\mathbf{f}_i^{пов.} = \delta_i \mathbf{t}_i = \sigma\kappa_i \mathbf{n}_i = -\sigma\nabla^2 c_i \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}.$$

Сила поверхневого натягу може бути застосована тільки до часток, що знаходяться біля або на поверхні рідини. Це обмеження, визначене чисельно через відношення поверхневої нормалі до її норми, стає нестабільним при наближенні норми до нуля. Єдиний спосіб уникнення чисельних проблем є введення певного граничного значення $\ell > 0$ концентрації частинок

$$\|\mathbf{n}_i\| \geq \ell.$$

Сила поверхневого натягу асиметрична за своїм задумом. У реальності ця сила є послідовністю взаємодії між молекулами води та повітря. Відсутність молекул повітря у моделювання не дає змоги зробити силу поверхневого натягу симетричною, через що і сама модель стає дещо віддаленою від дійсності.

Метод SPH є потужним інструментом, який зменшує складність математичних рівнянь потоків рідини, але, оскільки він спочатку був розроблений для завдань стисливого потоку, відсутність нестисливості є одним з основних недоліків методу. Особливості SPH, які роблять його ефективним обчислювальним алгоритмом, в кінцевому рахунку обумовлені тим, що він може бути отриманий з Лагранжіану і має властивості збереження Лагранжевої системи. Лагранжев

підхід вирішує труднощі, пов'язані з відсутністю симетрії великих пустот, набагато ефективніше, ніж Ейлерові методи. В результаті природним чином слідують збереження імпульсу і енергії разом з наближеним інваріантом циркуляції. Однак SPH також зберігає структуру, тобто кожна частинка несе свій склад незмінним, якщо матеріал, який представляє частинка, не піддається хімічним перетворенням. Ще одна приваблива особливість методу полягає в тому, динамічний діапазон просторової роздільної здатності регулюється плавно зі зміною щільності. У той же час цей метод має неперевершені властивості збереження не тільки енергії та імпульсу, але й кутового моменту. Також SPH є інваріантним та нечутливим до будь-яких похибок адвекції.

Немає ніяких обмежень, що накладаються ні на геометрію системи, ні на те, наскільки вона може еволюціонувати від початкових умов, так що початкові умови можуть бути легко запрограмовані без необхідності складних алгоритмів сітки, що використовуються в методах кінцевих елементів.

Завдячуючи його безсітковій природі, метод гідродинаміки згладжених частинок немає ніяких обмежень, що накладаються ні на геометрію системи, ні на те, наскільки вона може еволюціонувати від початкових умов, так що початкові умови можуть бути легко запрограмовані без необхідності складних алгоритмів сітки, що використовуються в методах кінцевих елементів.

Проте основним недоліком даного методу є його обмежена точність у багатовимірних потоках. Одне з джерел шуму породжується у апроксимаціях локальних інтерполянтів ядра через дискретні суми ближніх сусідів. Рух часток у багатовимірних просторах має значну більшу ступінь свободи, тому взаємновідштовхуючих сил високого тиску між сусідніми парами частинок не легко позбутися одночасно у всіх просторах. Як наслідок, деяке «тремтіння» у русі частинок, що при пересуванні швидко розростається у загальний шум.

Зокрема проблемою методу гідродинаміки згладжених частинок є те що метод інтерполяції, дуже простий, і на нього сильно вплине розлад частинок. SPH дає розумні результати для градієнтів першого порядку, але вони можуть бути гірше для похідних більш високого порядку. Іноді необхідно використову-

вати спеціальні методи при включенні похідних вищого порядку.

2.5 Архітектура графічного процессору та опис CUDA

CUDA – це комплект для розробки програмного забезпечення (SDK), який дозволяє запрограмовувати графічні карти від компанії Nvidia для різних обчислювальних завдань за допомогою мови програмування C++, з розширеннями для управління взаємодії між графічним процессором та хост-машиною. Крім драйверів пристроїв та посібника з програмування, SDK містить приклади кодів.

Відеокарти із підтримкою CUDA оснащені багатоядерними обчислювальними блоками як з локальною, так і зі спільною пам'яттю. За термінологією CUDA, одна графічна карта надає багатопроцесорні (потокові) мультипроцесори (SM), кожен з яких складається з декількох процесорів.

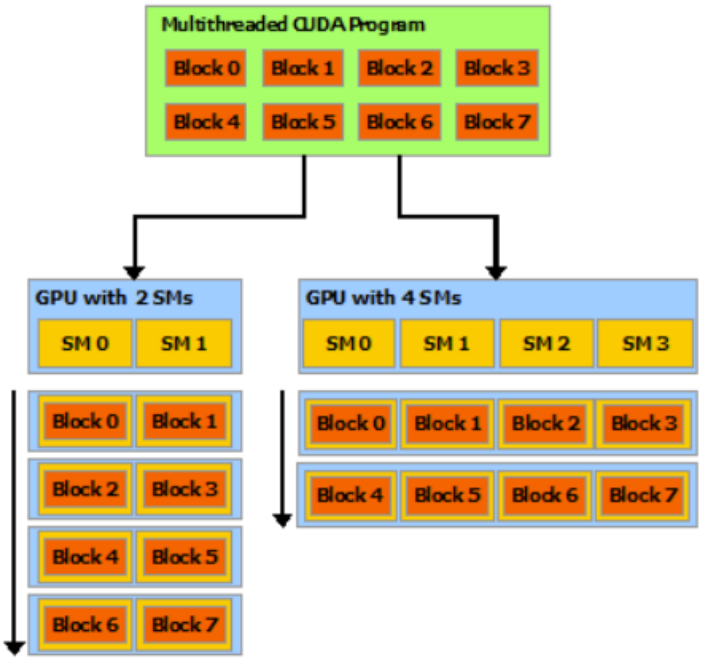


Рисунок 2.8 – Приклад обчислювальних блоків

Існує обмеження на кількість обчислювальних ниток на блок, оскільки всі

нитки блоку повинні знаходитися в одному ядрі процесора і повинні спільно використовувати обмежені ресурси пам'яті цього ядра. На поточних графічних процесорах блок потоків може містити до 1024 ниток.

Однак ядро може виконуватися за допомогою декількох рівномірно розташованих ниткових блоків, так що загальна кількість ниток дорівнює кількості нітей на блок, помноженому на кількість блоків.

Блоки організовані в одновимірну, двовимірну або тривимірну сітку ниткових блоків, як показано на рисунку 2.9. Кількість блоків ниток у сітці зазвичай диктується розміром оброблюваних даних, який зазвичай перевищує кількість процесорів у системі.

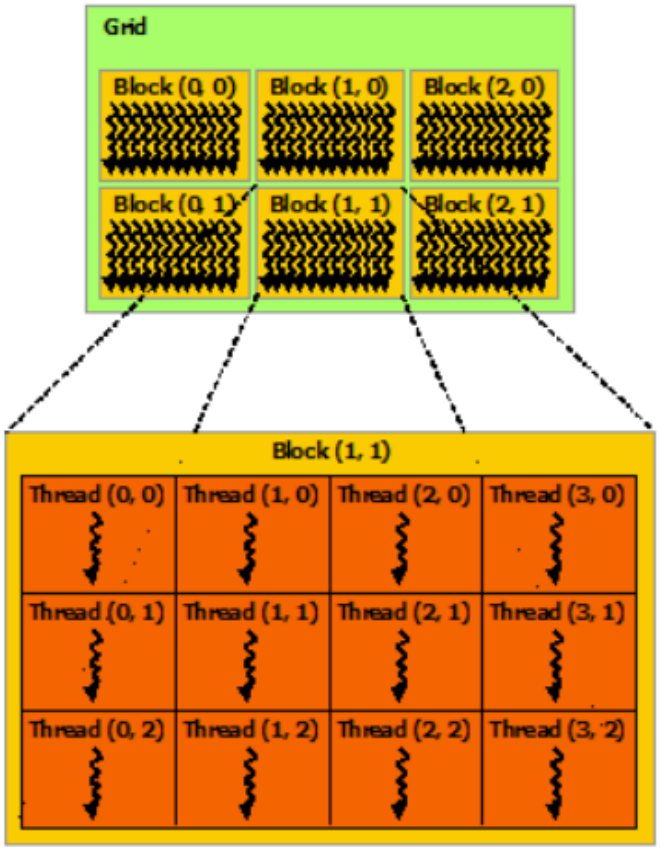


Рисунок 2.9 – Сітка обчислювальних блоків

Пам’ять, якою діляться всі SM, називається глобальною пам’яттю, а пам’ять, що ділиться процесорами в кожній SM - спільною пам’яттю. Існують додаткові простори пам'яті, такі як константа та пам'ять текстур, які доступні

лише для читання та спільного користування всіма SM, і забезпечують різні функції (наприклад, дані в текстурній пам'яті можуть бути інтерпольовані в апаратному забезпеченні).

Основна пам'ять GPU і CPU реалізована в DRAM, а кеш-пам'ять реалізована в SRAM із меншою затримкою. Структура сховища GPU і CPU в основному однакова.

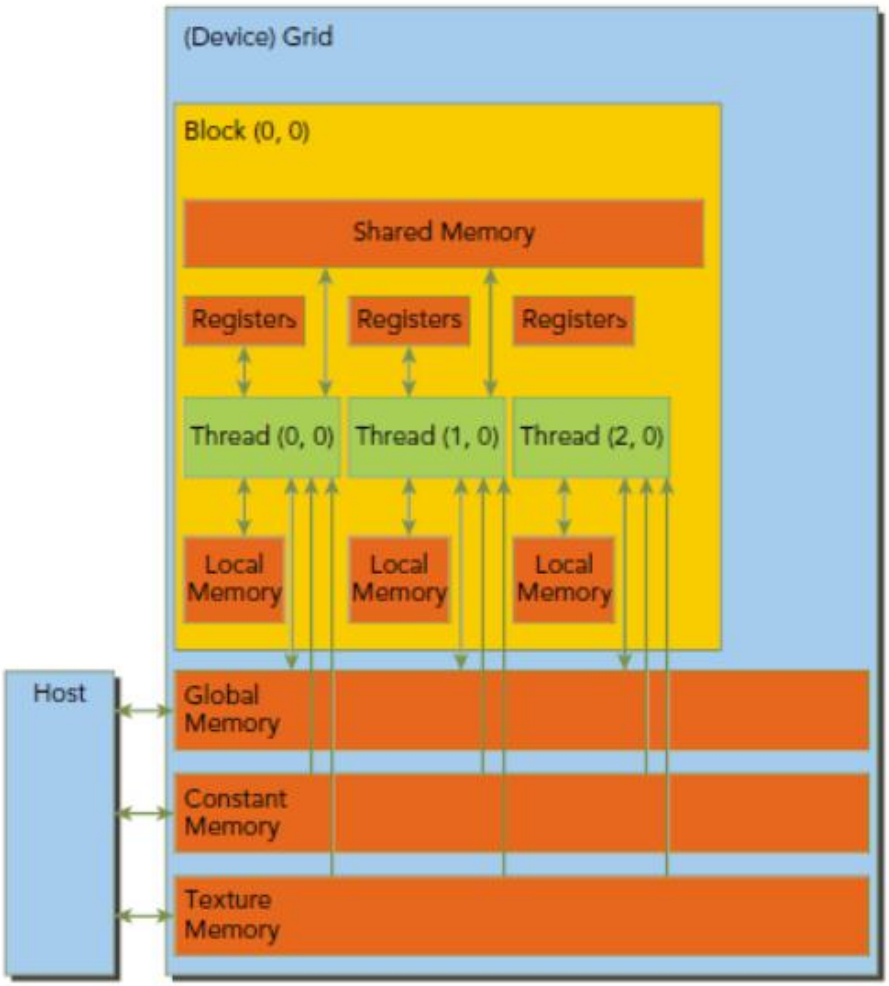


Рисунок 2.10 – Структура пам’яті CUDA

На наступному рисунку 2.11 показано взаємозв'язок передачі даних між процесором та графічним процесором. Можна помітити, що швидкість передачі даних між процесором та графічним процесором є відносно низькою (технологія NVLink може збільшитись у 5 – 10 разів), а швидкість передачі пам'яті GPU та вбудованої пам'яті значно вища. Тому для програмування завжди

розглядають можливість зменшення передачі даних між процесором і графічним процесором.

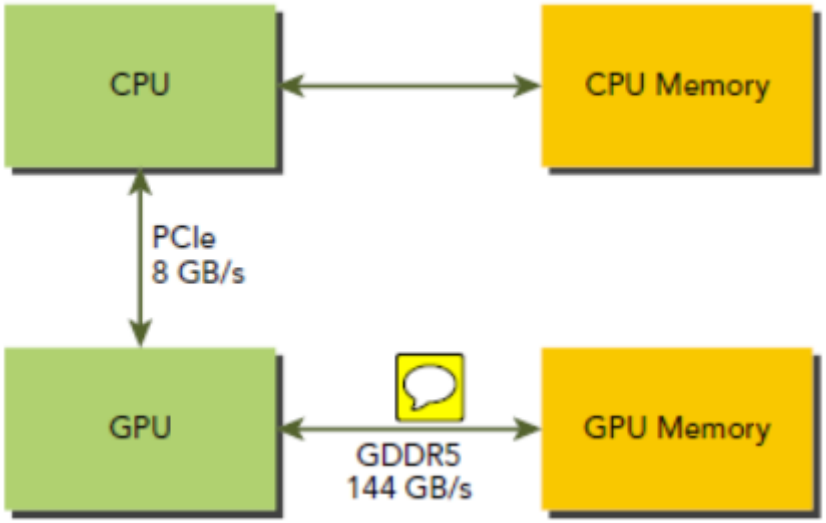


Рисунок 2.11 – Взаємозв'язок передачі даних між процесором та графічним процесором

Оскільки, CUDA використовується для обчислення, генерації даних, маніпулювання зображеннями, а OpenGL використовується для малювання пікселів або вершин на екрані та швидкість передачі даних між процесором та графічним процесором є відносно низькою ми використали Interop технологію яка дозволяє використовувати одну й ту ж спільну пам’ять загальну пам’ять у буфері кадрів графічного процессору для обчислень даних (CUDA) та для малювання зображення (OpenGL), що значно оптимізує весь процесс. OpenGL зберігає дані в абстрактних загальних буферах, які називаються буферними об'єктами. Взаємодія CUDA/OpenGL використовує одну просту концепцію: зіставити/розпакувати буфер OpenGL у простір пам'яті CUDA. Цей концепт зображений на рисунку 2.12.

Процесори можуть працювати паралельно, самостійно виконуючи ті ж самі набори операції з різними наборами даних. Таким чином, програмування в CUDA включає розробку ядер та визначення їх (паралелізованого) шаблону виконання. Тому модель виконання CUDA особливо добре підходить для методу

SPH, де для кожної частинки повинні виконуватися ті ж самі набори операцій (обчислення сили, швидкість та оновлення положення).

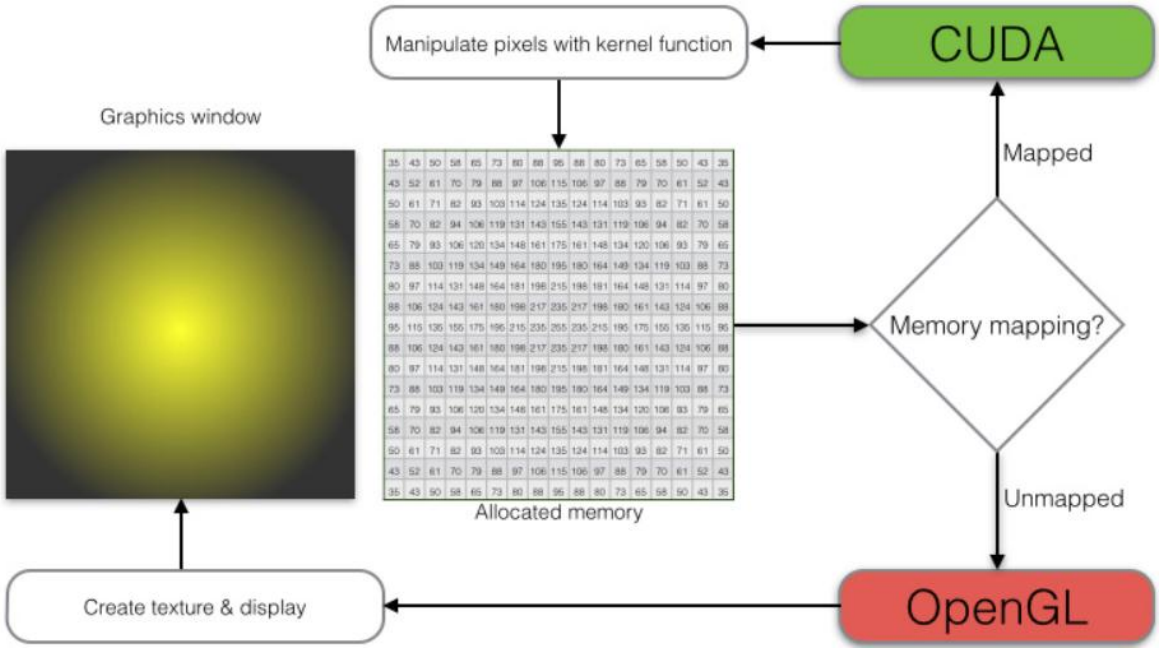


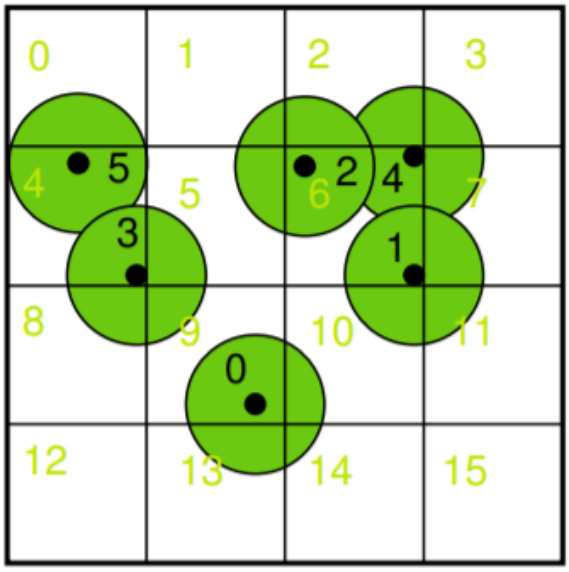
Рисунок 2.12 – Взаємозв'язок OpenGL та CUDA за допомогою Interop

2.6 Оптимізація обчислень на графічному процессорі

Дотримуючись класичного підходу SPH, алгоритм кроку часу моделювання може бути складений у наступні чотири кроки: пошук сусідів та оновлення списку, обчислення фізичних сил, оновлення швидкості та оновлення позиції.

Опишемо алгоритм пошуку сусідніх частинок. Тільки частинки, розташовані поруч одна з одною, вносять свій вклад у розрахунки сили в рівняннях руху. Якщо N - число частинок у системі, використання лише сусідів у розрахунках зводить проблему взаємодії $N * N$ до набагато меншої задачі порядку N. Справді, метод сортування частинок, який ми використовуємо, має порядок N, а кількість взаємодій, що розглядається на частинки, є константою на багато порядків менше, ніж N. Для пошуку сусідів ми використовуємо алгоритм, опи-

саний у прикладі "Particles" набору інструментів CUDA[12]. Згідно з якого обчислювальна область розділена на сітку лінійно індексованих комірок; кожній частинці присвоюється індекс клітини, в якому вона знаходиться, як хеш-значення, а частинки впорядковуються за своїм хеш-значенням. Потім список сусідів для кожної частинки будується шляхом пошуку сусідів в сусідніх комірках.



Cell id	Count	Particle id
0	0	
1	0	
2	0	
3	0	
4	2	3, 5
5	0	
6	3	1, 2, 4
7	0	
8	0	
9	1	0
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	

Рисунок 2.13 – Приклад сітки лінійно индексованих комірок

У побудові списку сусідів задіяні чотири ядра CUDA: обчислення хешу, сортування хеш-таблиці, впорядкування частинок та побудова списку сусідів. Структура даних сітки генерується з нуля на кожному часовому кроці. Цей вибір вимагає більше пам'яті, але призводить до більш швидкого виконання, оскільки пошук сусідів є операцією, пов'язаною з пам'яттю, і тому швидкість пошуку не збільшується з обчислювальною потужністю графічного процесора. Оскільки динамічний розподіл пам'яті неможливий в CUDA, вся пам’ять, необхідна для утримання максимальної кількості сусідів на частинку, помножена на кількість частинок, повинна бути виділена при ініціалізації. Дві версії пошуку сусідів представлені в [12], одна з яких використовує атомарні операції, представлених в останніх продуктах CUDA, операції дозволяють різним паралельним

потокам оновлювати одну і ту ж область пам'яті без конфлікту, що дозволяє створювати список сусідів на основі кожної комірки, а не частки.

Альтернативний підхід, який не вимагає атомних операцій - використання сортування. Алгоритм складається з декількох ядер. Перше ядро обчислює хеш-значення для кожної частинки на основі її ідентифікатора клітини. У цьому прикладі ми просто використовуємо лінійний ідентифікатор комірки як хеш, але це може бути корисним використання інших функцій, таких як крива Z - порядку, яка може бути використана для поліпшення узгодженості доступу до пам'яті. Ядро зберігає результати в масиві у глобальній пам'яті. Потім ми сортуємо частинки на основі їх хеш-значень. Сортування здійснюється за допомогою швидкого алгоритму сортування за розрядами, що забезпечується бібліотекою CUDPP. Це створює список ідентифікаторів частинок у порядку комірок. Для того, щоб цей відсортований список був корисним, нам потрібно мати можливість знайти початок будь-якої даної комірки в відсортованому списку. Це досягається запуском іншого ядра, яке використовує обчислювальну ніть графічного процессору для кожної частинки та порівнює індекс клітини поточної частинки з індексом клітини попередньої частки у відсортованому списку. Якщо індекс відрізняється, це вказує на початок нової комірки, а початкова адреса - записана в інший масив з використанням розсіяного запису.

У обчисленні фізичних сил задіяні два ядра CUDA: обчислення густини, та обчислення внутрішніх сил для кожної частинки. В обчисленні внутрішних сил, кожна частинки використовує окрему обчислювальну ніть, в якій спочатку обчислюється список сусідів, після чого обчислюються внутрішні сили частинки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Використання C++ та GLUT для моделювання течії в'язкої рідини та для побудови тривимірного геометричного об'єкта

Для проведення розрахунків та графічного зображення результатів було обрано об'єктно-орієнтовану мову програмування C++, виходячи з наступних міркувань.

По-перше, масштабованість. Проект був реалізований в об'єктно-орієнтованій парадигмі програмування, що розширює рамки проекту, дозволяючи додавати нові методі, функціонал, більш складні форми об'єктів без корінних змін в вже існуючому проекті. Окрім цього, масштабованість самої мови програмування дозволяє користуватися програмою на різних платформах та операційних системах.

По-друге, обчислювальна швидкодія. Більша частка програм, які потребують ефективних і швидких розрахунків написані на мові C або C++. Швидкодія досягається, тим що вихідний код під час компіляції відразу переводиться у машинний, та саме у C++ операції з числами, з плаваючою комою є найшвидшими.

По-третє, узагальнення типів даних для алгоритмів. У C++ дослідникам надається можливість створення алгоритмів для різних типів даних. Завдяки цьому були створені шаблони для роботи класів Point та Volume, де зміні можуть бути задані типами даних double, float або int. Таким чином, пристосованість до роботи з різними типами даних дозволяє змінювати структурні особливості моделі без корінних змін в структури проекту.

Також слід зазначити про можливість використання препроцесорних директив, макросів та простору імен, завдяки чому отримаємо потужний інструмент розробки та вдосконалення програмних продуктів.

Графічна складова проекту реалізована за допомогою графічної бібліотеки GLUT, яка є прямим спадкоємцем оригінальної відкритої графічної бібліоте-

ки OpenGL, що визначає незалежний від мови програмування кросплатформний програмний інтерфейс (API), для візуалізації 2D та 3D комп’ютерної графіки.

Головними відмінностями GLUT є:

– відкритий вихідний код;

– підтримка розробниками, актуальність версії;

– єдиний спрощений апаратно-незалежний інтерфейс;

– вбудована реалізація таких складних геометричних примітивів як куб, шар, диск тощо;

– підтримка кросплатформеності, що гарантує однаковий візуальний результат незалежно від типу операційної системи та організації відображення інформації.

Функції GLUT реалізовані в моделі клієнт-сервер. Додаток виступає в ролі клієнта – воно виробляє команди, а сервер GLUT інтерпретує і виконує їх. Сам сервер може знаходитися як на тому ж комп'ютері, на якому знаходиться клієнт (наприклад, в виді динамічної бібліотеки – DLL), так і на іншому (при цьому може бути використаний спеціальний протокол передачі даних між машинами).

GLUT обробляє і малює в буфері графічні примітиви. Кожен примітив – це точка, відрізок, багатокутник і т.д. Кожен режим може бути змінений незалежно від інших. Визначення примітивів, вибір режимів і інші операції описуються за допомогою команд у формі викликів функцій прикладної бібліотеки.

З точки зору архітектури графічна система GLUT є конвеєром, що складається з декількох послідовних етапів обробки графічних даних.

Команди GLUT завжди обробляються в тому порядку, в якому вони надходять, хоча можуть відбуватися затримки перед тим, як проявиться ефект від їх виконання.

3.2 Використання CUDA для оптимізації моделювання течії в’язкої рідини

Для оптимізації розрахунків було обрано комплект для розробки програмного забезпечення CUDA, тому що, вона об'єднує кілька речей:

– інтеграція з обладнанням, яке може бути використане для паралельних обчислень, з відповідними драйверами для цього;

– мова програмування, заснована на мові програмування C, для програмування зазначеного обладнання, та мова для збірки, яку інші мови програмування можуть використовувати як ціль;

– набір для розробки програмного забезпечення, що включає бібліотеки, різні засоби налагодження, профілювання та компіляції, а також прив'язки, що дозволяють мовам програмування на стороні процесора викликати код на стороні графічного процесора.

Суть CUDA полягає в написанні коду, який може працювати на сумісних масивно паралельних архітектурах SIMD: сюди входить кілька типів графічних процесорів, а також апаратне забезпечення, що не є графічним процесором, таке як NVIDIA Tesla. Розроблене для паралельних обчислень апаратне забезпечення може виконувати значно більшу кількість операцій в секунду, ніж центральний процесор, при досить подібних фінансових витратах, приводячи до поліпшення продуктивності в 50 разів і більше в ситуаціях, що це дозволяють.

Однією з переваг CUDA перед іншими методами є те, що вона доступна як мова загального призначення, замість того, щоб використовувати піксельні та вершинні шейдери для програмування комп'ютерів загального призначення. Ця мова базується на C з кількома додатковими ключовими словами та концепціями, що полегшує розуміння для програмістів, які не мають досвіду роботи з графічном процессором.

Також перевагаю є легкий старт, щоб почати використовувати CUDA, потрібно завантажити SDK, прочитати інструкцію і мати сумісне обладнання CUDA для того щоб почати свої перши обчислення на графічному процессорі.

3.3 Опис програми

Існуюча структура програми складається з наступних компонентів класів:

– Cuboid – клас, що будує куб;

– Volume – клас, що будує межу області на основі методу *R* -функцій;

– ROperations – клас, що визначає R-функції;

– Shapes – клас, що містить різні рівняння фігур, побудовані з використанням методу R-функцій.

– MarchingCubes – клас, що реалізує алгоритм крокуючих кубиків.

– Point – клас, що зберігає координати точки в робочій області;

– Particle – клас, що описує фізичні характеристики частинки;

– Collisions – клас, що перевіряє частинки на зіткнення між собою та з межею області;

– NeighbourSearch – клас, що описує алгоритм пошуку сусідніх частинок;

– Forces – клас, що описує фізичні сили, які діють на частинки;

– Integrator – клас, що описує метод чисельного інтегрування Верле;

– Renderer – клас, що графічно відображає процес моделювання;

– SPH – клас, що ініціалізує процес моделювання.

Опишемо структуру версії программи яка оптимізована для обчислень на графічному процессорі:

– Sph_system.cpp – C++ файл, що містить ініціалізацію системи моделювання;

– Sph_system.cu – C файл, в якому знаходиться код обчислювальних ядер графічного процессору.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ

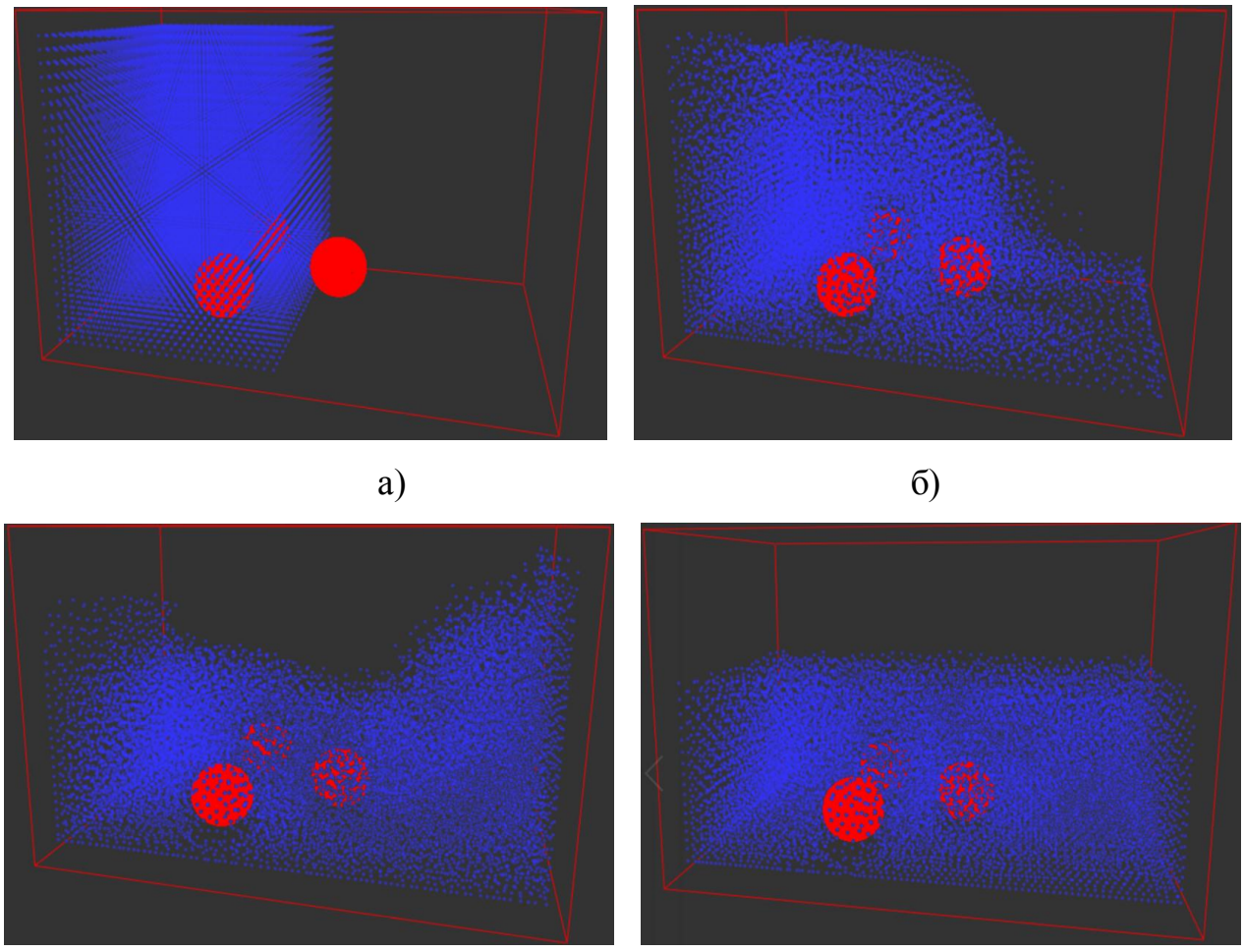
Обчислювальний експеримент був проведений у кубічній області розмірами 1 кубічна умовна одиниця. Максимальна кількість модельованих частинок сягнула 16000, для кожної було задано значення фізичного параметра, що відповідає характеристиці води. У таблиці 4.1 наведені відповідні параметри.

Таблиця 4.1 – Набір значень фізичних параметрів для моделювання води

Опис	Символ	Значення	Одиниці
Густина	ρ	998.29	$\frac{\text{кг}}{\text{м}^3}$
Маса	m	0.02	кг
В’язкість	μ	3.5	$\text{Па} \cdot \text{с}$
Поверхневий натяг	σ	0.0728	$\frac{\text{Н}}{\text{м}}$
Порогова товщина	l	7.065	–
Коефіцієнт пружності	k	3.0	$\frac{\text{Н}}{\text{м}}$
Коефіцієнт повернення	c_R	0	–
Радіус згладжування	h	0.0457	м

Також в області моделювання було побудовано три геометричних об’єкта – циліндри. Було ініціалізовано кількість частинок в вигляді стіни води, яка під впливом сили тяжіння падає на фігури та повинна зіштовхнутися з геометричними об’єктоми та змінити свою структуру відповідним чином. Результати екс-

перименту зображені на рисунках 4.1 – 4.2.



а) б)

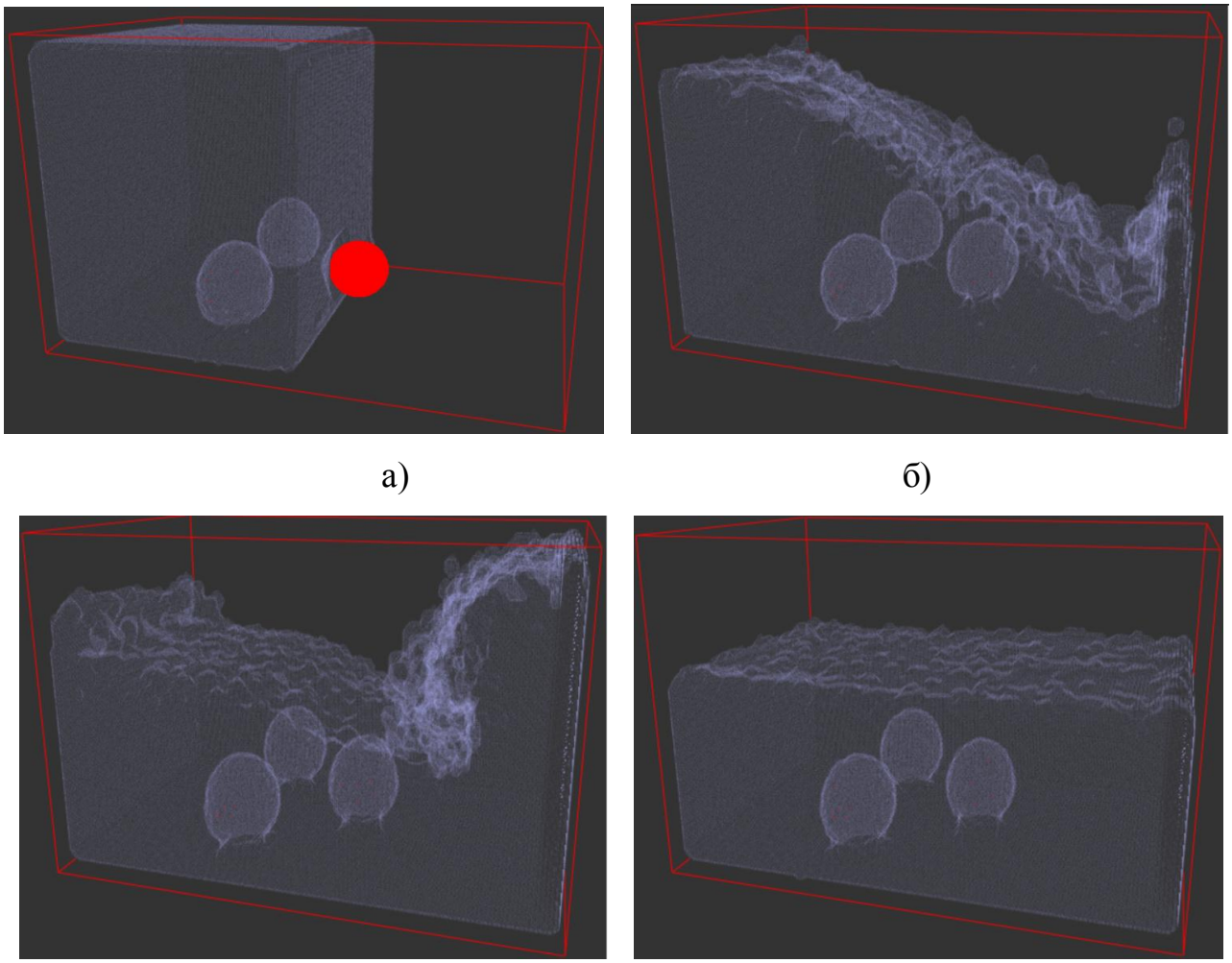
в) г)

а) на першому кроці; б) на другому кроці;

в) на третьому кроці; г) на четвертому кроці

Рисунок 4.1 – Результати експерименту в вигляді окремих частинок

З результатів обчислень, реалізація CUDA на графічному процесорі мала дуже низькі показники порівняно з версією на головному процесорі при низьких рівнях кількість частинок. Це пов'язано із накладними витратами на обчислення та сортування сусідів, пошук сусіднього сегмента та роботу на ядрах, які є індивідуально набагато повільніше, ніж центральний процесор. Впровадження CUDA не починає перевершувати показники версії центрального процесора, поки у вас не буде близько 1000 частинок, після чого накладні витрати на пошук сусідів забезпечують достатню вигоду.



а) б)

в) г)

а) на першому кроці; б) на другому кроці;

в) на третьому кроці; г) на четвертому кроці

Рисунок 4.2 – Результати експерименту в вигляді води

Пошук сусідів забезпечує CUDA версію коду лінійною масштабованістю, це означає, що до того часу як ви отримаєте до 15 000 частинок реалізація на графічному процесорі перевершує реалізацію на центральному процесорі майже в 100 разів. Основним вузьким місцем у реалізації на центральному процессорі є сортування, яке в кінцевому підсумку приймає більшість часу при збільшенні кількості частинок і має складність $N * N$, ніж реалізація на графічному процесорі зі складність N.

Таблиця 4.2 – Порівняння результатів симуляції на центральному процесорі та графічному процесорі

Кількість частинок	Кількість кадрів в секунду на центральному процессорі	Кількість кадрів в секунду на графічному процессорі
100	2073.8	401.2
1000	646.3	297.5
5000	34.6	222.2
10000	8.7	171.4
15000	3.9	140.5
50000	0.8	59.8

5 АНАЛІЗ МОЖЛИВИХ ЗАСТОСУВАНЬ

Оптимізована модель течії в’язкої нестисливої рідини може знайти своє застосування у різних сферах, пов’язаних із гідродинамічними процесами та моделюванням в реальному часі. Прикладом може бути відкритий для публіки в 2009 році причал Понт-дель-Петролі який зазнав кількох штормів, деякі з яких завдали серйозних збитків та подальшої потреби в великих ремонтних роботах. Зокрема, морська буря Глорія залишила причал сильно пошкодженим, змусивши місцеву владу закрити доступ громадськості до нього. Важко охарактеризувати хвильовий клімат в районі, що оточує Понт-дель-Петролі, але висота хвиль може досягати 5-6 метрів. Метод гідродинаміки згладжених частинок може бути використаний для генерації хвиль, та допомогти глибше зрозуміти і дослідити механізм, що призвів до пошкоджень, а метод R-функцій побудувати складний геометричний об’єкт, як причал. Також імплементація гідродинаміки згладжених частинок на графічному процесорі може бути широко використана у моделюванні зсувів під час виверження вулканів, де лава може бути представлена як дискретна кількість частинок, а складна поверхня склонів вулканів, може тоже бути побудована за домогою методу R-функцій.

На основі описаної концепції роботи з моделями їх застосування є актуальним також у галузях промислової аеродинаміки, балістики, ливарного виробництва, кораблебудування та авіабудування тощо. Моделювання динамічних об’єктів активно використовується також і у дослідницьких наукових сферах астрофізики, вулканології та океанографії, а саме: дослідження космічних об’єктів та небесних тіл, вивчення підземних течій та поведінки виверження вулканів, прогнозування льодовикових танень і заводнення регіонів світу.

ВИСНОВКИ

Під час виконання атестаційної роботи було оптимізовано метод моделювання руху течії в’язкої нестисливої рідини з перешкодами у тривимірному просторі на основі методу гідродинаміки згладжених частинок та ду R-функцій за допомогою виконання обчислень на графічному процессорі.

Результати обчислювального експерименту дозволяють сказати, що високопродуктивна обчислювальна архітектура, надана графічними процесорами з підтримкою CUDA, ідеально підходить для методу гідродинаміки згладжених частинок. У завданнях, що вимагають великих обчислювальних витрат, ми досягаємо лінійної масштабованості щодо кількості обчислювальних одиниць в графічному процесорі, що демонструє оптимальність реалізації. Використання графічних карт та CUDA дозволило отримати багато детальних деталей з прикладних проблем фізики, дякуючи можливості запуску обчислень із сотнями разів більшої кількості частинок за набагато коротший час, ніж це потрібно для аналогічного коду на одному процесорі або навіть на багатьох кластерах. Це прискорення робить моделювання на основі методу гідродинаміки згладжених частинок більш доступним та корисним для досліджень. Тому цей метод може бути застосований до різних задач моделювання рідини, де розглядається виявлення зіткнень з довільними об'єктами та потрібні результаті в реальному часі. Подальші дослідження можуть бути присвячені, оптимізації виконання всіх обчислень у блоці графічної обробки або врахуванню інших фізичних величин.

Оскільки нелінійні диференціальні, що описують поводження течії в’язкої нестисливої рідини, в загальному випадку неможливо описати в аналітичному вигляді, то досить актуальною на сьогоднішній день залишається проблема пошуку і використання чисельних методів, які спростять пошук розв’язку задачі з заданою точністю. Комбінація методів, яка використовувалась в роботі, є одним з безлічі ефективних поєднань, але не є найкращим, тому подальші дослідження у галузі чисельних методів можуть бути присвячені створенню нових підходів та видів функціональної взаємодії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Математическое моделирование технологических физических процессов, численное моделирование CFD. URL : http://www.cfdgroup.ru (дата звернення: 06.04.2019).

2. Расчет гидродинамических воздействий на возвращаемый аппарат при посадке на воду / В. В. Жаркова, А. Е. Щеляев, А. А. Дядькин [та ін.] // Компьютерные исследования и моделирование. 2017. С. 37–46.

3. Paul W. Cleary, Joseph Ha. Three-dimensional smoothed particle hydrodynamics simulation of high pressure die casting of light metal components // Journal of Light Metals. 2002. V. 2, N. 3. P. 169–183.

4. Kolb A., Cuntz N. (2005). Dynamic particle coupling for GPU-based fluid simulation // Proc. 18th Symposium on Simulation.

5. Harada T., Koshizuka S., Kawaguchi Y. Smoothed particle hydrodynamics on GPUs // Proc. Comput. Graph. Intl. 2007. P. 63–70.

6. Amada T., Imura M., Yasumuro Y., Manabe Y., Chihara K. Particle-based fluid simulation on the GPU. 2003. 58 p.

7. Ландау Л. Д., Лифшиц Е. М. Теоретическая физика. Гидродинамика. Москва : Наука, 1986. 736 с.

8. Кравченко В. Ф., Рвачев В. Л. Алгебра логики, атомарные функции и вейвлеты в физических приложениях. Москва : Физматлит, 2006. 416 с.

9. Рвачев В. Л. Теория R-функций и некоторые ее приложения. Киев : Наукова думка, 1982. 552 с.

10. Kelager M. Lagrangian fluid dynamics using smoothed particle hydrodynamics. University of Copenhagen, 2006. 88 p.

11. Dobek S. Fluid Dynamics and the Navier-Stokes Equitation. University of Maryland, 2012. 14 p.

12. Green, S., NVIDIA: CUDA Particles, Presentation slides. Tech. rep., NVIDIA. 2008.

13. NVIDIA CUDA C++ Programming Guide. URL :

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (дата звернення: 29.10.2020).