МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кваліфікаційна наукова
праця на правах рукопису

КУЗЬМИЧ ВАЛЕНТИН АНАТОЛІЙОВИЧ

УДК 004.032.26 (043.3)

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ РУХУ РІДИН З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

123 Комп’ютерна інженерія
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело_____

Науковий керівник: Новотарський Михайло Анатолійович, д. т. н., професор

Київ – 2023

АННОТАЦІЯ

Кузьмич В.А. Методи та засоби математичного моделювання руху рідин з використанням машинного навчання. - Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп'ютерна інженерія з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2023.

Дисертаційна робота присвячена розробці дворівневого методу моделювання руху рідин на основі решітчастої моделі Больцмана та згорткової нейронної мережі, що дозволяє точно і ефективно моделювати рух нестисливих рідин.

Решітчаста модель Больцмана - це математичний інструмент, який знаходить широке застосування в чисельному моделюванні руху газів та рідин. Вона ґрунтується на статистичних принципах та дозволяє моделювати рух частинок в рідині на основі їхньої взаємодії через взаємодію з іншими частинками та перешкодами. Решітчаста модель Больцмана дозволяє враховувати мікроскопічні взаємодії частинок та отримувати макроскопічні властивості рідини, такі як тиск, температура та швидкість.

Машинне навчання - це область науки використовує алгоритми та моделі, які дозволяють комп'ютерам навчатися на даних та робити прогнози або приймати рішення без явного програмування. У контексті дослідження руху рідини, машинне навчання може бути використане для аналізу великих обсягів даних та побудови прогностичних моделей.

Згорткові нейронні мережі є одним з типів моделей машинного навчання, які знайшли широке застосування в різних практичних сферах діяльності, таких як обробці зображень, включаючи аналіз руху рідини. Вони імітують спосіб, яким працює візуальний кортекс у людей, дозволяючи автоматично визначати особливості та закономірності в даних. Згорткові нейронні мережі здатні виявляти

взаємозв'язки між частинами зображень та ефективно використовувати цю інформацію для розв'язання завдань, пов'язаних із рухом рідини.

Поєднання обчислювальної гідромеханіки і машинного навчання відіграє ключову роль у вирішенні актуальних проблем і завдань в багатьох галузях науки та техніки. В різних наукових та технологічних сферах людської діяльності існують величезні вимоги до точності та ефективності моделювання руху рідин, особливо в важливих галузях, таких як аеродинаміка, морська гідродинаміка, автомобільна і космічна інженерія, біомедицина і багато інших. Розуміння та передбачення поведінки рідини є важливим елементом для оптимізації дизайну, підвищення продуктивності і зменшення витрат в цих галузях.

Використання решітчастої моделі Больцмана у поєднанні з машинним навчанням відкриває нові можливості для точного та швидкого моделювання руху рідин. Такий підхід дозволяє знижувати обчислювальну складність та споживання ресурсів, що є критичним для великих і складних задач гідродинаміки. Особливо важливим є використання машинного навчання для автоматичного аналізу великих обсягів даних, отриманих під час моделювання, і для здатності виділяти з них ключові закономірності, які можуть бути важливими для подальшого удосконалення моделі.

Запропоновано дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана і згорткової нейронної мережі, що використовується для уточнення значень поля швидкостей на основі вирішення крайової задачі на основі рівняння Пуассона, який відрізняється від відомих методів тим, що зменшує час моделювання.

Набув подальшого розвитку метод решітчастої моделі Больцмана за рахунок методу розпаралелювання на основі підходу domain decomposition та використання модифікованої рівноважної функції розподілу на основі мінімізації дискретної ентропії, яка відрізняється від відомих методів кращою безумовною лінійною стабільністю моделювання.

Набув подальшого розвитку метод моделювання розв’язку крайової задачі на основі рівняння Пуассона для тиску, значення якого використовуються для

корекції поля швидкості при моделюванні нестисливих рідин, на основі модифікованої нейронної мережі, що враховує геометрію обчислювального простору, який відрізняється від відомих методів можливістю обробки складних обчислювальних областей та обчислювальною швидкістю.

Запропоновано адаптацію дворівневого методу моделювання руху рідини для використання на спеціальному обчислювальному пристрої, яка відрізняється тим, що забезпечує зменшення кількості обчислень для розробленої нейронної мережі при моделюванні розв’язку крайової задачі на основі рівняння Пуассона для тиску.

У першому розділі здійснено огляд наукової літератури по тематиці досліджень. Розглянуті та описати основні типи методів моделювання руху рідин. Було виділено три основні типи підходів до моделювання руху рідин: методи моделювання на основі рівняння Нав'є-Стокса, методи LBM та методи машинного навчання. Був описаний їх загальний історичний розвиток, наведено загальні переваги та недоліки цих типів методів. Також окремо було розглянута література про методи розв'язання рівняння Пуассона, яке є важливою складовою в кожному з трьох згаданих вище підходах. Завдяки аналізу результатів першого розділу була сформована задача дисертаційного дослідження: розробка дворівневого методу моделювання руху потоку рідини, за допомогою методу LBM та машинного навчання.

В другому розділі описується метод решітчастої моделі Больцмана: його місце в контексті рівнів абстракції опису рідини, теоретичне обгрунтування можливості застосування методу LBM для моделювання руху рідин та зв’язок між рівнянням Больцмана на рівнянням Нав’є-Стокса. Розглянуті та описані найбільш поширені чисельні схеми. Описані механізми задання початкових та граничних умов, що використовуються в методі LBM. Були розглянуті особливості задання притоку та витоку рідини, умови зворотного відображення, що моделює взаємодію потоку рідини з твердим тілом. Була описана модифікована рівноважна функція розподілу на основі мінімізації дискретної ентропії, що дозволяє досягти безумовну лінійну стабільність моделювання. Обгрунтовано необхідність уточнення поля

швидкості за допомогою рівняння Пуассона для тиску під час моделювання руху нестисливих рідин методом LBM.

В третьому розділі дисертації було досліджено особливості використання нейронних мереж для моделювання розв’язку крайової задачі на основі рівняння Пуассона. Оскільки нейронні мережі можуть задавати набагато складніші функції джерела, ніж традиційні аналітичні або чисельні методи, а також через чисельну ефективність нейронних мереж, Використання нейронних мереж для моделювання розв’язку рівняння Пуассона є обіцяючим підходом. Також були розглянуті ітераційні чисельні методи вирішення систем алгебраїчних рівнянь, які традиційно використовуються для моделювання розв’язку рівняння Пуассона. Доцільне їх використання для генерації навчального та тестового датасетів для нейронної мережі. Були розглянуті різні шари штучних нейронних мереж та функції активацій, які використовуються для досягнення бажаних результатів. Приведений загальний огляд процесу вирішення диференціальних рівнянь за допомогою штучних нейронних мереж.

В четвертому розділі був розроблений дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі. Була описана структура нейронної мережі для моделювання розв’язку крайової задачі на основі рівняння Пуассона, особливості генерації тренувального датасету для нейронної мережі, розглянуті особливості приведення результатів роботи нейронної мережі до значень тиску, який використовується для корекції значень поля швидкості в методі LBM. Детально описаний алгоритм дворівневого методу моделювання руху рідини, що складається з методу LBM, який використовує модифіковану рівноважну функцію розподілу, та з розробленої згорткової нейронної мережі для моделювання розв’язку крайової задачі на основі рівняння Пуассона. Розроблений масштабований паралельний алгоритм для дворівневого методу на основі підходу domain decomposition. Розглянуті види апаратних прискорювачів для штучних нейронних мереж. На основі їх переваг і недоліків, нейронна мережа була оптимізована під обраний прискорювач NPU. Для тестування розробленого методу було розроблене тестове програмне забезпечення

для моделювання руху рідин в довільних обчислювальних просторах розміру 96×96, з можливістю зміни параметрів моделювання.

В п’ятому розділі був проведений аналіз результатів моделювання руху рідин з допомогою розробленого дворівневого методу, який був реалізований в тестовому програмному забезпеченні. Результати експериментів підтвердили здатність розробленого дворівневого методу забезпечувати нестисливість рідини, в порівнянні зі звичайним методом LBM. Була досліджена точність нейронної мережі в порівнянні з чисельним методом. Була показана взаємна відповідність між ними. Були досліджені обчислювальна швидкість розробленого методу та вплив використання різних апаратних прискорювачів на швидкість обчислень. Дворівневий метод показав більш ніж у 6 разів кращу ефективність, ніж чисельний метод, при використанні GPU, та у 13 разів кращу ефективність, при використанні NPU.

Розроблений метод дозволяє точно і ефективно моделювати рух нестисливих рідин та використовувати різні типи апаратних прискорювачів для збільшення швидкості обчислень згорткової нейронної мережі. Це має практичне значення для різних областей наукової та технічної діяльності, зокрема біомедична інженерія, моделювання руху рідин в гідротехнічних конструкціях.

Ключові слова: математичне моделювання, комп’ютерне моделювання, обчислювальна гідродинаміка, рівняння Пуассона, рівняння Нав’є-Стокса, решітчаста модель Больцмана, машинне навчання, нейронна мережа, згорткова нейронна мережа, регресія, спеціалізовані обчислювальні пристрої, нейронні процесори.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці в яких опубліковано основні наукові результати дисертації:

1. Novotarskyi M.A., Stirenko S.G., Gordienko Y.G., **Kuzmych V.A.** Deep reinforcement learning with sparse distributed memory for “Water World” problem solving //Radio Electronics, Computer Science, Control.– 2021.– № 1.– P.136-143. DOI 10.15588/1607-3274-2021-1-14
2. **Kuzmych V.A.**, Novotarskyi M.A., Nesterenko O.B. Solving Poisson Equation with Convolutional Neural Networks // “Radio Electronics, Computer Science, Control” – 2022.– № 1. p. 48-57. DOI 10.15588/1607-3274-2022-1-6
3. **Valentyn Kuzmych**, Mykhailo Novotarskyi, Accelerating simulation of the PDE solution by the structure of the convolutional neural network modifying, Lecture Notes on Data Engineering and Communications Technologies 135, p. 3-15 ISSN 23674512 DOI 10.1007/978- 3-031-04809-8 (Scopus)
4. М.А.Новотарський Дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі / М.А. Новотарський, **В.А. Кузьмич** // Електронне моделювання – Т45, № 5 – Київ, 2023. -С. 39-53

Праці наукових конференцій:

1. **Kuzmych V.**, Novotarskyi M. “Application of machine learning in the modeling of physical processes” // International Conference ICSFTI2020, May 14-15, 2020. Зроблено доповідь
2. **Kuzmych V.**, Novotarskyi M. Solving Poisson Equation with Convolutional Neural Networks // International Conference ICSFTI2021, May 12-13, 2021. Зроблено доповідь.
3. **Valentyn Kuzmych**, Mykhailo Novotarskyi, Accelerating simulation of the PDE solution by the structure of the convolutional neural network modifying, The 2nd International Conference on Artificial Intelligence and Logistics Engineering (ICAILE2022) - Virtual Conference, 20-22 February 2022. Зроблено доповідь.

4. **Kuzmych V.**, Novotarskyi M. Simulation of fluid motion in closed surfaces using a lattice Boltzmann model // International Conference ICSFTI2022, June 30, 2022. Зроблено доповідь.

ABSTRACT

Kuzmych V.A. Methods and means of mathematical modeling of fluid flow using machine learning. - Qualifying scientific work on manuscript rights.

Dissertation for the degree of Doctor of Philosophy in the specialty 123 - Computer Engineering and 12 - Information Technologies. - National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2023.

The dissertation work is devoted to the development of a two-level method for modeling the movement of fluids based on the lattice Boltzmann model and a convolutional neural network, which allows for accurate and efficient modeling of the movement of incompressible fluids.

Lattice Boltzmann model is a mathematical tool that is widely used in the numerical modeling of the movement of fluids. It is based on statistical principles and allows modeling the movement of particles in a liquid based on their interaction through interactions with other particles and obstacles. Lattice Boltzmann model allows to take into account the microscopic interactions of particles and obtain the macroscopic properties of the fluid, such as pressure, temperature and velocity.

Machine learning is a field of science that uses algorithms and models that allow computers to learn from data and make predictions or make decisions without explicit programming. In the context of fluid motion research, machine learning can be used to analyze large volumes of data and build predictive models.

Convolutional neural networks are one of the types of machine learning models that have found wide application in various practical fields, such as image processing, including fluid motion analysis. They mimic the way the human visual cortex works, allowing for automatic detection of features and patterns in data. Convolutional neural networks are able to detect relationships between parts of images and effectively use this information to solve problems related to fluid movement.

The combination of computational hydromechanics and machine learning plays a key role in solving current problems and tasks in many fields of science and technology. In various scientific and technological fields of human activity, there are huge

requirements for the accuracy and efficiency of fluid motion modeling, especially in important fields such as aerodynamics, marine hydrodynamics, automotive and space engineering, biomedicine, and many others. Understanding and predicting fluid behavior is an essential element to optimize design, improve productivity and reduce costs in these industries.

The use of the lattice Boltzmann model in combination with machine learning opens up new opportunities for accurate and fast modeling of fluid motion. This approach allows to reduce computational complexity and resource consumption, which is critical for large and complex problems of hydrodynamics. Especially important is the use of machine learning for the automatic analysis of large volumes of data obtained during modeling and for the ability to extract from them key regularities that can be important for further improvement of the model.

The two-level method for modeling fluid motion using a lattice Boltzmann model and a deep neural network, which is used to refine the values of the velocity field based on the solution of a boundary value problem based on the Poisson level, was proposed. Developed method differs from known methods in that it decreases the time of simulation.

The Boltzmann lattice model method was further developed through the parallelization method based on the domain decomposition approach and the use of a modified equilibrium distribution function based on discrete entropy minimization, which differs from known methods by better unconditional linear stability of modeling.

The method of modeling the solution of the boundary value problem based on the Poisson equation for pressure, the value of this material for the correction of the velocity field in the modeling of carrier fluids, based on a modified neural network that takes into account the geometric computing space, was further developed. Developed, which differs from known methods, providing ease of processing complex computational areas and computational speed.

The adaptation of the two-level method of fluid motion simulation for use on a special computing device is proposed, which is distinguished by the fact that it provides an acceleration of the calculation speed for the developed neural network when simulating the solution of the boundary value problem based on the Poisson equation for pressure.

In the first chapter, a review of the scientific literature on the topic of research is carried out. Considered and described the main types of fluid movement simulation methods. Three main types of fluid motion modeling approaches have been identified: modeling methods based on the Navier-Stokes equation, LBM methods, and machine learning methods. Their general historical development was described, and the general advantages and disadvantages of these types of methods were given. The literature on methods for solving the Poisson equation, which is an important component in each of the three approaches mentioned above, was also separately reviewed. Thanks to the analysis of the results of the first chapter, the dissertation research task was formed: the development of a combined method of modeling the flow of liquid, using the LBM method and machine learning.

The second chapter describes the method of the lattice Boltzmann model: its place in the context of the abstraction levels of the fluid description, the theoretical justification of the possibility of using the LBM method for modeling the movement of fluids, and the connection between the Boltzmann equation and the Navier-Stokes equation. The most common numerical schemes are considered and described. Mechanisms for setting initial and boundary conditions used in the LBM method are described. The peculiarities of setting the inflow and outflow of liquid, the conditions of reverse reflection, which simulates the interaction of the flow of liquid with a solid body, were considered. A modified equilibrium distribution function based on discrete entropy minimization was described, which allows achieving unconditional linear stability of the simulation. The need to refine the velocity field using Poisson's equation for pressure during modeling the movement of incompressible fluids by the LBM method is substantiated.

In the third chapter of the dissertation, the peculiarities of using neural networks for solving the boundary value problem based on the Poisson equation were investigated. Because neural networks can handle much more complex source functions than traditional analytical or numerical methods, and because of the numerical efficiency of neural networks, using neural networks to solve the Poisson equation is a promising approach. Iterative numerical methods for solving systems of algebraic equations, which are traditionally used to solve Poisson's equation, were also considered. It is appropriate

to use them for the generation of training and test datasets for the neural network. Different layers of artificial neural networks and activation functions used to achieve the desired results were considered. A general overview of the process of solving differential equations using artificial neural networks is given.

In the fourth chapter, a two-level method of fluid motion simulation was developed using the lattice Boltzmann model and a convolutional neural network. The structure of the neural network for solving the boundary value problem based on the Poisson equation was described, the features of generating a training dataset for the neural network, the features of bringing the results of the neural network to the pressure values, which is used to correct the values of the velocity field in the LBM method, were considered. The algorithm of the two-level fluid motion simulation method consisting of the LBM method, which uses a modified equilibrium distribution function, and the developed convolutional neural network for solving the boundary value problem based on the Poisson equation is described in detail. A scalable parallel algorithm for the two-level method based on the domain decomposition approach is developed. Considered types of hardware accelerators for artificial neural networks. Based on their advantages and disadvantages, the neural network was optimized for the selected NPU accelerator. To test the developed method, a test software was developed for modeling the movement of fluids in arbitrary computational spaces of size 96×96, with the possibility of changing the modeling parameters.

In the fifth chapter, the analysis of the results of fluid movement modeling was carried out using the developed two-level method, which was implemented in the test software. The results of the experiments confirmed the ability of the developed two-level method to ensure the incompressibility of the liquid, in comparison with the usual LBM method. The accuracy of the neural network was investigated in comparison with the numerical method. A mutual correspondence between them was shown. The computational speed of the developed method and the influence of the use of various hardware accelerators on the computational speed were investigated. The two-level method showed more than 6 times better performance than the numerical method when using GPU and 13 times better performance when using NPU.

The developed method makes it possible to accurately and efficiently simulate the movement of incompressible fluids and use various types of hardware accelerators to increase the speed of convolutional neural network calculations.

Key words: mathematical modeling, computer modeling, computational fluid dynamics, Poisson's equation, Navier-Stokes equation, lattice Boltzmann model, machine learning, neural network, convolutional neural network, regression, specialized computing devices, neural processors.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LBM - Lattice Boltzmann methods

CNN – convolutional neural network

ReLU - Rectified Linear Unit

GPU - graphics processing unit

NPU - Neural Processing Unit

FEM – finite element method

LGA – lattice gas automata

LBE – Lattice Boltzmann equation

СЛАР – система лінійних алгебраїчних рівнянь

ВСТУП

Актуальність теми. Математичне моделювання складних фізичних процесів має велике значення в різних областях науки і техніки. Він передбачає формулювання математичних рівнянь або моделей, які описують поведінку складних систем, що дозволяє нам зрозуміти, передбачити та проаналізувати їхні властивості. Воно забезпечує структурований і кількісний підхід до вивчення складних фізичних явищ, що веде до глибшого розуміння, покращення дизайну, оптимізації систем і прийняття обґрунтованих рішень у багатьох наукових та інженерних дисциплінах.

Одним із таких класів явищ є рух потоків рідини. Методи обчислювальної гідромеханіки є розділом механіки рідини, що займається чисельним моделюванням і аналізом процесів потоку рідини. Як інструмент, такі методи використовуються в інженерних і наукових дослідженнях для вивчення та прогнозування поведінки рідин, наприклад рідин і газів, у різних сценаріях.

Методи обчислювальної гідромеханіки базуються на математичних рівняннях, які описують збереження маси, імпульсу та енергії для моделювання потоку рідини. Ці рівняння, відомі як рівняння Нав'є-Стокса, розв'язуються за допомогою чисельних методів і алгоритмів. Поділяючи область рідини на невеликі контрольні об'єми або комірки, методи обчислювальної гідромеханіки дискретизують рівняння та розв'язують їх ітераційно, щоб отримати наближені рішення. Але такий підхід на основі чисельного розв'язання рівняння Нав'є-Стокса має кілька недоліків, і найголовнішим з яких є обчислювальна складність такого підходу. Рівняння передбачають розв'язування великої кількості пов'язаних диференціальних рівнянь із частковими похідними, що призводить до інтенсивного обчислювального моделювання. Як наслідок, тривалий час моделювання та високі обчислювальні витрати можуть бути обмежуючим фактором, особливо для великомасштабного моделювання або застосування у реальному часі.

Альтернатвним підходом до моделювання руху рідин є решітчаста модель Больцмана. Для моделювання обчислювальної області використовується сітка, де комірки розглядаються як крупні частинки. Однак, замість використання рівнянь

Ейлера або Нав'є-Стокса для опису поведінки цих частинок, використовується кінетичне рівняння Больцмана. Характеристики цих крупних частинок статистично описуються за допомогою функції розподілу, яка враховує координати та швидкості частинок. Проте застосування цього методу є проблематичним для моделювання нестисливих рідин. Можливим рішенням є використання дворівневих схем, що були досліджені в роботах таких дослідників як Чен, Дулен, Новотарський, Інамуро та ін.

Використання машинного навчання для моделювання руху рідин є актуальним в останні роки, в зв'язку з наступними факторами: використання машинного може прискорити моделювання руху рідин порівняно з традиційними числовими методами; моделі машинного навчання, такі як нейронні мережі, можуть працювати паралельно та використовувати оптимізовану обчислювальну архітектуру, таку як графічні процесори (GPU) або спеціалізовані обчислювальні пристрої (наприклад, тензорні процесори), для ефективного обчислення значень тиску; використання машинного навчання дозволяє легко модифікувати та адаптувати модель для різних умов та граничних умов.

Вищезгадані проблеми визначають актуальність теми дисертації, а також її практичне, наукове значення.

Зв’язок роботи з науковими програмами, планами, темами. Наукові дослідження виконувались в рамках договору від 14.02.2023 №0123U101109? який укладений у рамках робіт за грантом НАТО “Наука заради миру та безпеки”, що виконується на кафедрі обчислювальної техніки, Національного технічного університету України (Київський політехнічний інститут імені Ігоря Сікорського).

Мета і завдання дослідження. Метою цієї дисертаційної роботи зменшення часу моделювання дворівневим методом моделювання руху нестисливої рідини на основі решітчастої моделі Больцмана та згорткової нейронної мережі, і його практичного втілення у програмному забезпеченні та спеціалізованому обчислювальному пристрої для подальшого практичного застосування у біомедичній інженерії.

Для реалізації поставленої мети необхідно вирішити такі задачі:

- Дослідити методи моделювання руху рідин;
- Розробити дворівневий метод типу предиктор-коректор для моделювання руху нестисливих рідин.
- Розробити згорткову нейронну мережу для моделювання процесу розв’язання крайової задачі на основі рівняння Пуассона
- Адаптувати нейронну мережу для використання її за допомогою спеціалізованого обчислювального пристрою.
- Створити програмне забезпечення для моделювання руху нестисливих рідин.

Об’єкт дослідження – процес моделювання руху нестисливих рідин в закритих просторах з використанням машинного навчання.

Предмет дослідження – методи та засоби математичного моделювання руху нестисливих рідин в закритих просторах з використанням машинного навчання.

Методи дослідження. Методами дисертаційного дослідження були аналіз та систематичний розбір літературних джерел та теоретичних викладок, що присвячені методам моделювання руху рідин та особливостям застосування машинного навчання при моделюванні фізичних процесів. У роботі використані чисельні методи моделювання, методи машинного навчання, методи решітчастої моделі Больцмана. Для реалізації розробленого методу були використані сучасні підходи та технології розробки програмного забезпечення та використання апаратних прискорювачів.

Наукова новизна отриманих результатів:

- *Запропоновано* дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана і згорткової нейронної мережі, що використовується для уточнення значень поля швидкостей на основі моделювання розв’язку рівняння Пуассона, який відрізняється від відомих методів тим, що зменшує час моделювання.
- *Набув подальшого розвитку* метод решітчастої моделі Больцмана за рахунок методу розпаралелювання на основі підходу domain decomposition та використання модифікованої рівноважної функції розподілу на основі

мінімізації дискретної ентропії, яка відрізняється від відомих методів кращою безумовною лінійною стабільністю моделювання.

- *Набув подальшого розвитку* метод моделювання розв'язку рівняння Пуассона для отримання розподілу тиску з подальшим його використанням для корекції поля швидкості при моделюванні нестисливих рідин, на основі модифікованої нейронної мережі, що враховує геометрію обчислювального простору, який відрізняється від відомих методів можливістю обробки складних обчислювальних областей та меншим часом обчислення.
- *Запропоновано* адаптацію дворівневого методу моделювання руху рідини для використання на спеціальному обчислювальному пристрої, яка відрізняється тим, що забезпечує прискорення обчислень для розробленої нейронної мережі при моделюванні розв'язку крайової задачі на основі рівняння Пуассона для тиску.

Практичне значення отриманих результатів. Результати дисертаційного дослідження дозволяють застосувати розроблений метод моделювання руху нестисливих рідин в різних сферах інженерної та наукової діяльності людини, де швидкість і точність моделювання є важливими факторами. Розроблений метод дозволяє моделювати рух рідин в обчислювальній області з довільним розташуванням твердого тіла та різноманітними граничними умовами притоку та витоку рідини. Розроблений метод може бути прискорений за допомогою різних типів апаратних прискорювачів, таких як GPU та NPU. Зважаючи на перебування України в стані війни, актуальним сьогодні є застосування розробленого методу в сфері біомедичної інженерії, оскільки велика кількість людей, в порівнянні з мирним часом, потребує якісної медичної допомоги та лікування, які в багатьох випадках вимагають точного та швидкого моделювання роботи внутрішніх органів людини.

Особистий внесок здобувача. Дисертаційна робота є самостійною науковою працею. Основні результати та наукові положення дисертаційної роботи отримані здобувачем самостійно у процесі науково-дослідної роботи. В роботах, опублікованих у співавторстві, дисертанту належать: [1] - реалізація коду

програми, яка виконувала моделювання; [2] - розробка структури нейронної мережі для розв’язання рівняння Пуассона, формування тренувального та тестового датасетів, розробка програмного коду; [3] - реалізація експериментів по дослідженню швидкодії нейронної мережі; [4] – розробка дворівневого методу моделювання руху рідин, розробка тавиконання експериментів**.**

Апробація результатів дисертації. Основні результати роботи опубліковано та обговорено на міжнародних конференціях: “The International Conference on Security, Fault Tolerance, Intelligence” (м. Київ, 2020, 2021, 2022 р.); The 2nd International Conference on Artificial Intelligence and Logistics Engineering, ICAILE 2022 (м. Київ, 2022).

Публікації. За результатами досліджень було опубліковано 4 наукові статті, 2 з яких входять до наукометричної бази даних з міжнародним індексом цитування Web of Science, 1 – Scopus.

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, п’яти розділів, загальних висновків, списку використаних джерел із 237 найменувань та додатків. Загальний обсяг дисертації становить 213 сторінок, з яких 127 сторінок основного тексту, 2 додатки на 49 сторінках, та містить 51 рисунок, 129 формул, 3 таблиці.

РОЗДІЛ 1
ОГЛЯД МЕТОДІВ МОДЕЛЮВАННЯ ДИНАМІКИ РІДИН

1.1 Методи моделювання на основі рівняння Нав'є-Стокса

Рівняння Нав'є-Стокса, виведене із законів збереження маси, імпульсу та енергії, керують поведінкою рідин. Це рівняння можна перевести у еквівалентну систему диференціальних рівнянь у часткових похідних, які описують поля швидкості, тиску та густини рідини. Розв'язання цих рівнянь має вирішальне значення для розуміння явищ потоку рідини, таких як аеродинаміка, гідродинаміка та теплопередача. Рівняння Нав'є-Стокса для рідини у векторній формі виглядає наступним чином:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \vec{F} \tag{1.1}$$

де $\vec{u} = \vec{u}(\vec{r}, t)$ - швидкість рідини, $\vec{r}$ - радіус-вектор точки простору, $t -$ час, ∇ - оператор Гамільтона, ρ - густина, $p = p(\vec{r}, t) -$ тиск, $\vec{F}$ – сторонні сили.

Специфічна нелінійність рівнянь Нав'є-Стокса, наявність малого параметра в багатьох випадках при старшій похідній та просторовий нестаціонарний характер руху робить знаходження загального розв'язання рівняння надзвичайно складною проблемою. Існування і гладкість рішень рівнянь Нав'є - Стокса - одне з семи математичних завдань тисячоліття, сформульованих в 2000 Математичним інститутом Клея [232]. Ці обмеження значно ускладнюють аналітичне дослідження цих рівнянь, і відомо небагато випадків аналітичного інтегрування рівнянь Нав'є-Стокса [2,3] та призводять до висновку, що єдиним ефективним способом розв'язку рівняння Нав'є-Стокса є застосування чисельних методів.

Однією із перших робіт, де було запропоноване чисельне рішення рівнянь Нав'є-Стокса стала робота [1]. Цей метод відомий як метод «штучної стисливості». В рівняння нерозривності автор ввів додатковий член зі штучною стисливістю, що протягом установлення рішення по часу, наближується до нуля:

$$\frac{\partial p}{\partial t} + c^2 \nabla \cdot \mathbf{u} = 0 \qquad (1.2)$$

де c^2 – довільна константа. Основним недоліком такого методу є нестабільність при великих значеннях c^2.

Подальшим розвитком методу «штучної стисливості» став метод проекцій, розроблений, незалежно один від одного, у роботах [5] та [6]. Основна перевага методу проекції полягає в тому, що обчислення полів швидкості та тиску відбуваються незалежно одне від одного. Зазвичай метод проекції використовує кілька кроків обчислення для кожного числового кроку в часі. У багатьох алгоритмах, що базуються на методі проекції, кроки поділяються таким чином:

1. Спочатку система просувається в часі до положення середнього кроку в часі, розв'язуючи рівняння переносу маси та імпульсу за допомогою відповідного методу адвекції. Це називається крок предиктора.
2. У цей момент початкова проекція може бути реалізована таким чином, що поле швидкості в середині кроку в часі визначається як вільне від дивергенції.
3. Потім виконується крок корекції алгоритму. Вони використовують значення швидкості, щільності тощо, що були обчислені на середині часового кроку, для формування остаточного стану кроку в часі.
4. Потім застосовується остаточна проекція для забезпечення обмеження розбіжності на поле швидкості.

Наступним етапом у розвитку чисельних методів розв'язання рівняння Нав'є-Стокса став метод «маркерів та комірок» [4]. Цей підхід відрізняється тим, що вихідні рівняння формулюються в термінах змінних "швидкість-тиск", а для побудови кінцево-різницевої схеми використовується рознесена сітка. Цей метод дискретизує простір на кубічні комірки шириною h. Кожна комірка має тиск p, визначений у її центрі. Він також має швидкість, $\mathbf{u} = (u_x, u_y, u_z)$, але компоненти швидкості розміщені в центрах трьох граней комірки, , u_x на грані x-min, u_y на грані y-min, і u_z на грані z-min, як показано на рис. 1.1. Розташування швидкості

таким чином веде до більш стабільних симуляцій, ніж збереження швидкості в центрах комірок.

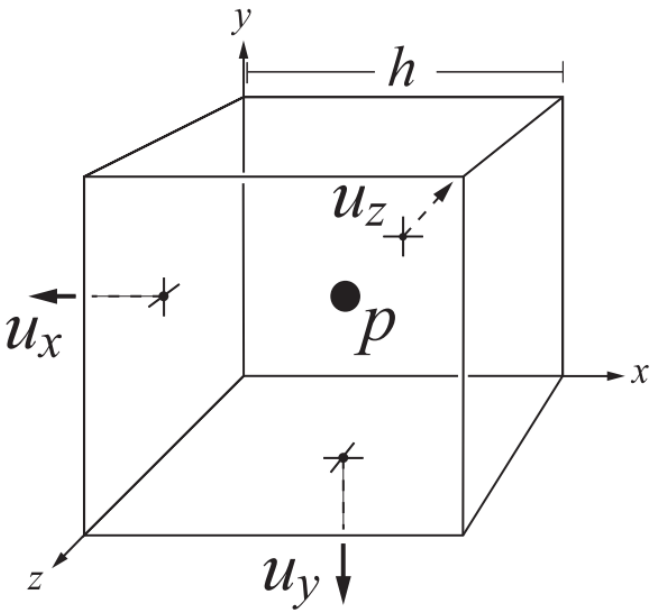

Рисунок 1.1 – Кубічна комірка методу «маркетів і комірок». Компоненти швидкості, u_x, u_yі u_z, зберігаються на мінімальних гранях комірки. Тиск, *p*, зберігається в центрі комірки.

Також слід відзначити метод «змінних напрямів» [7,8]. Суть метода полягає в дискретизації рівняння в просторі та часі за допомогою підходу кінцевих різниць або кінцевих елементів. Область поділена на сітку, а час дискретизовано на часові кроки. Основна ідея методу полягає в тому, щоб почергово змінювати напрямок оновлення змінних. У цьому випадку невідома функція $\mathrm{u(x,y,t)}$ оновлюється по черзі вздовж напрямку *x* і *y*. Процес триває, доки розв'язок *u* не зійде до стаціонарного стану або не досягне бажаного рівня точності. Однак цей метод має кілька недоліків [9,10]:

1. Метод змінних напрямів вимагає повторення оновлень у напрямку *x* та *y*, доки не буде досягнуто збіжності. Цей ітераційний процес може збільшити обчислювальну складність порівняно з іншими методами, які вирішують всю систему одночасно.
2. Точність залежить від роздільної здатності сітки
3. Вибір коефіцієнтів *(α, β, γ)* у параболічному рівнянні може вплинути на властивості стабільності та збіжності методу

4. Метод в основному призначений для вирішення параболічних рівнянь у двох просторових вимірах. Розширення його до вищих розмірів може бути складним через підвищену складність і вимоги до обчислень.

Метод «маркерів і комірок» став основою для роботи [11]. Розроблений метод отримав назву «метод крупних частинок». Використовуючи нерухому ейлерову сітку, обчислювальна область поділяється на комірки, які розглядаються як крупні частинки. Рух цих крупних частинок моделюється за допомогою нестаціонарної моделі Ейлера. Таким чином, метод посередній між методом частинок у комірках Харлоу і звичайними кінцево-різницевими підходами, оскільки використовуються закони збереження для комірки скінченого розміру, але без подальшого граничного переходу до точки.

Метод крупних частинок володіє широкими обчислювальними можливостями, що дозволяють досліджувати складні потоки навколо тіл різної форми, як при дозвукових, так і надзвукових швидкостях. Основним недоілком методу є нестійкість чисельної схеми з ростом числа Рейнольдса до значень 10^2-10^3.

Також необхідно згадати групу методів, що отримали назву SIMPLE (Semi-Implicit Method for Pressure Linked Equations) [13,14,15]. Він особливо ефективний для проблем потоку нестисливої рідини. Метод SIMPLE складається з наступних кроків:

1) Обчислювальна область дискретизується на сітку, як правило, за допомогою методів кінцевого об’єму або кінцевих різниць. Керівні рівняння, а саме рівняння неперервності (збереження маси) та рівняння імпульсу (збереження імпульсу), дискретизуються на цій сітці.

2) Ключовою особливістю методу SIMPLE є сильний зв’язок між тиском і швидкістю. Рівняння корекції тиску вводиться, щоб гарантувати, що поле швидкості задовольняє умову відсутності розбіжності (нестисливість) і забезпечує збереження маси.

3) Рівняння корекції тиску отримують шляхом віднімання коригуючого члена з рівнянь імпульсу, що дозволяє роз’єднати поля тиску та швидкості.

Цей поправковий член формулюється на основі градієнта тиску та розбіжності поля швидкості. Отримане рівняння корекції тиску є рівнянням Пуассона для коригувального тиску.

4) Процедура вирішення:

a) Крок прогнозування: робиться початкове припущення для поля швидкості, а попереднє поле швидкості обчислюється шляхом нехтування членом градієнта тиску в рівняннях імпульсу.

b) Рівняння корекції тиску розв’язується для коригувального тиску за допомогою ітераційного методу. Коригувальний тиск отримується шляхом приведення розбіжності скоригованого поля швидкості до нуля.

c) Крок корекції: виправлене поле швидкості отримується шляхом віднімання градієнта коригувального тиску з попереднього поля швидкості.

d) Кроки b і c повторюються, поки поле швидкості та коригувальний тиск не збіжаться до рішення. Критерії збіжності можуть базуватися на визначеному допуску або максимальній кількості ітерацій.

5) Завершення: коли поле швидкості та коригувальний тиск збігаються, поле тиску оновлюється шляхом додавання коригувального тиску до початкового припущення тиску. Оновлене поле тиску можна використовувати для подальшого аналізу або пост-обробки.

Однак, цей метод має значущі недоліки – повільна збіжність в деяких випадках, неточне обчислення коригувального тиску, і, як наслідок, неточна корекція тиску [16]. Як наслідок, було запропоновано ряд методів, що покращують оригінальний алгоритм [17,18].

Розглянуті вище методи є теоретичною основою для спеціалізованого програмного забезпечення. Так, існує ряд високоефективних програмних продуктів, призначених для моделювання гідродинамічних явищ на основі рівнянь Нав'є-Стокса. Один із найбільш визнаних інструментів – OpenFOAM [234], який є відкритим програмним забезпеченням та має широкі можливості для розв'язання різноманітних гідродинамічних завдань.

Ще однією платформою є ANSYS Fluent [235], яка надає інтегровані рішення для моделювання руху рідини та теплообміну. Ця система використовує потужні чисельні методи для аналізу складних гідродинамічних систем у різних областях, таких як авіаційна та автомобільна промисловість, енергетика та біотехнології.

Крім того, SimScale [236] визначається як онлайн-платформа для інженерного моделювання, яка надає можливості для гідродинамічного аналізу через хмарні обчислення. Вона дозволяє інженерам та дослідникам здійснювати дослідження різних гідродинамічних сценаріїв, надаючи доступ до розширених інструментів і великого обчислювального потенціалу.

Також варто зазначити COMSOL Multiphysics [237], яка є многофізичною програмою для моделювання теплопередачі, механіки рідин, та електромагнетизму. Цей інструмент надає широкі можливості для гідродинамічного моделювання, особливо в тандемі з іншими фізичними явищами, що розширює область його застосування в різноманітних галузях.

1.2 Методи моделювання розв’язку рівняння Пуассона

Моделювання розв’язання рівняння Пуассона відіграє вирішальну роль у чисельному розв’язанні рівнянь Нав’є-Стокса, особливо для задач моделювання руху нестисливих рідин. Рівняння Пуассона має такий вигляд:

$$\Delta\varphi = f \tag{1.3}$$

де φ – невідома функція, Δ – оператор Лапласа, f – довільна функція.

У контексті рівнянь Нав’є-Стокса рівняння Пуассона зазвичай виводять із рівняння безперервності і використовують для отримання поля тиску.

Рівняння Пуассона для тиску виводиться шляхом врахування розбіжності рівнянь імпульсу та накладення умови нестисливості через рівняння нерозривності. Це призводить до рівняння Пуассона, яке пов’язує лапласіан тиску з джерелом, що включає розбіжність конвективного та дифузійного потоків. Моделювання розв’язку рівняння Пуассона дозволяє отримати поле тиску, яке необхідно для обчислення поля швидкості.

Оскільки рівняння Пуассона є диференційним рівнянням в часткових похідних, то в більшості випадках, вирішити його можна тільки з використанням чисельних методів. Ці методи діляться на типи по методу дискретизації рівняння та області його розв’язування. Найпоширенішими методами дискретизації рівняння Пуассона є:

1) метод скінчених різниць
2) метод скінченних елементів
3) безсіткові методи

Далі йде огляд застосування даних методів дискретизації при моделюванні розв’язку рівняння Пуассона.

1.2.1 Метод скінченних різниць

Метод скінченних різниць - це метод, що розбиває область, в якій вирішується рівняння, на сітку скінченних різниць. Диференціальні оператори замінюються на апроксимації скінченних різниць, і отримане дискретне рівняння розв'язується чисельно.

Одним із найпопулярніших методів розв’язання дискретного рівняння Пуассона, у вигляді системи лінійних алгебраїчних рівнянь, є метод Якобі. Так у роботі [19] автори пропонують метод Якобі поєднаний з нейроеволюційними алгоритмами для моделювання розв’язку рівняння Пуассона, що містить розриви в значеннях густини середовища. Варто відзначити, що недоліком цієї роботи є труднощі з інтерпретації результатів роботи штучного інтелекту.

В іншій роботі [20] автори використали метод Якобі як частину комплексного вирішення рівняння Пуассона. Проте експерименти обмежились областями з невеликою кількістю вузлів, на які була поділена обчислювальна область, не дозволяє говорити про можливість застосування запропонованого методу для моделювання областей з великою кількістю дискретизованих точок.

Також варто відмітити роботу [21], де автори продемонстрували можливість застосування методу Якобі для розв’язання рівняння Пуассона в одно- та

двовимірному просторах, використовуючи графічні прискорювачі зі спільною пам’яттю.

Іншою групою методів, які використовуються для розв’язання дискретного рівняння Пуассона, є точні методи розв’язування систем лінійних алгебраїчних рівнянь. Однією із перших робіт, де був застосований аналіз Фур’є як точний метод для розв’язання рівняння Пуассона була робота [22]. Розроблений метод продемонстрував значну точність та швидкодію, в порівнянні з ітераційними методами. Але це було досягнути зі значними обмеженнями – були застосовані прості періодичні граничні умови, незначна кількість вузлів, на які була поділена обчислювальна область, обмеження в обчислювальних ресурсах того часу.

Наступним кроком у розвитку точних методів стала робота [23]. Автор застосував точні методи для розв'язання рівняння Пуассона в непрямокутній області, та порівняв з ітераційними методами.

Однією із перших робіт по застосуванню точних методів для вирішення рівняння Пуассона в тривимірному просторі стала робота [24]. Автори продемонстрували ефективність аналізу Фур'є для тривимірних випадків. Також автори вказують на основні недоліки їх роботи – моделювання лише граничних умов Неймана, без граничних умові Діріхле чи періодичних граничних умов.

Ще одним широко використовуваним ітераційним методом розв'язку системи лінійних рівнянь є метод Гауса-Зейделя [26], який є схожим до методу Якобі [25]. Однією із перших роботі, де був застосований метод Гауса-Зейделя для розв’язання рівняння Лапласа (частковий випадок рівняння Пуассона) стала робота [29]. Інколи метод Гауса-Зейделя називають «методом Лейбмана» [30]. Метод Лейбмана набув подальшого розвитку у роботах [31, 32].

Модифікацією метода Гауса-Зейделя став симетрчний метод Гауса-Зейделя [33,34].

У роботі [27] автори використали метод Гауса-Зейделя для розробки спеціального апаратного обчислювача для розв’язку рівняння Пуассона. До недоліків даної роботи можна віднести фіксований розмір обчислювальної області, що може бути оброблена розробленим обчислювачем.

Також слід відзначити метод адаптивного згладжувача Гауса-Зейделя [28] для рівняння Пуассона у двовимірному просторі. Модифікований метод продемонстрував більшу швидкодію, в порівнянні з класичним методом Гауса-Зейделя, але є менш точним.

Стабільність методу Гауса-Зейделя була досліджена в роботі [35] на прикладі розв’язання рівняння Пуассона в одновимірному просторі, та була порівняна з іншими ітераційними методами.

Методи релаксації [36] – окрема група методів розв’язання систем лінійних алгебраїчних рівнянь, які є варіантом методу Гауса-Зейделя. Одним із ключових проблем цього методу є підбір параметру релаксації. Вирішенням цієї проблеми можна відзначити працю [37] присвячену пошуку оптимального параметру релаксації для розв’язання рівняння Пуассона. Ключовою особливістю цієї роботи є теоретична можливість пошуку параметру релаксації для простору довільної розмірності.

У роботі [38] автори розробили ефективну схему для розпаралелювання методу релаксації на графічних прискорювачах на інших спеціалізованих обчислювачах. Для тестування методу вони використали рівняння Пуассона в тривимірному просторі.

Подальшого розвитку метод релаксації набув в роботі [39]. В цій роботі розв'язок рівняння Пуассона був використаний для обробки графічних зображень. Розроблений метод показав кращу швидкодію, ніж оригінальний метод релаксації.

1.2.2 Метод скінченних елементів

Метод скінченних елементів – це метод, що поділяє область на скінченну кількість підобластей, відомих як скінченні елементи. Рівняння Пуассона розв'язується на кожному елементі, а потім використовується апроксимація, яка забезпечує неперервність розв'язку на границях елементів. Цей метод дозволяє більш точно моделювати геометрію складних областей.

Однорідна крайова задача Діріхле для рівняння Пуассона та відповідне моделювання її розв’язання є популярною темою досліджень [40]. Були розглянуті рішення для трикутних та полігональних елементів, на які розбивається двовимірна обчислювальна область. Також в цій роботі розглянутий метод Гальоркіна [41] як основа для методу кінцевих елементів.

Серед більш прикладних робіт слід відзначити роботу [42], в якій автори використали метод скінченних елементів для розв’язання рівняння Пуассона, для обчислення станів електронів в квантових наноструктурах.

У роботі [43] досліджується застосування методів скінченних елементів (FEM) для розв’язування рівняння Пуассона та його різноманітні застосування. Стаття, охоплює математичні основи скінченних елементів, включаючи дискретизацію області, побудову базисних функцій кінцевих елементів і формулювання кінцевої лінійної системи рівнянь.

Крім того, стаття,, заглиблюється в чисельну реалізацію конечних елементів, обговорюючи такі теми, як складання матриці жорсткості, накладення граничних умов і методи вирішення для отриманої лінійної системи.

Автори роботи [44] розробили паралельну реалізацію методу скінченних елементів для рівняння Пуассона в тривимірному просторі. Вони провели дослідження, що підкреслює важливість тривимірного моделювання для малих геометричних пристроїв. Метод використовує неструктуровану тетраедральну сітку та застосовує метод скінченних елементів із специфічним формулюванням для вузлів, розташованих на межі між областями з різними характеристиками. Формулювання враховує різні властивості матеріалу з обох боків межі розділу. Методи декомпозиції домену та мультикомп’ютери з розподіленою пам’яттю використовувались для вирішення пов’язаних лінійних систем із застосуванням бібліотеки MPI [45] в цій роботі. Ця робота дає змогу моделювати такі ефекти, як термоемісійне випромінювання та тунелювання, розв’язуючи рівняння Пуассона та електронно-діркові рівняння.

1.2.3 Безсіткові методи

Безсіткові методи, такі як метод фундаментальних рішень і методи радіальної базисної функції, не вимагають попередньо визначеної сітки. Ці методи використовують розсіяні точки даних і функції інтерполяції для наближення рішення. Рівняння Пуассона перетворюється на систему алгебраїчних рівнянь, які можна розв’язувати ітераційно або за допомогою інших методів. Безсіткові методи є вигідними для задач зі складною геометрією та нерегулярними областями.

Однією із перших робіт, де розглядається застосування безсіткових методів для розв’язання рівняння Пуассона, є робота [46]. В своїй роботі автори розробили метод який назвали «методом дифузійних елементів», та який має кілька переваг в порівнянні з метод скінченних елементів. Цей метод отримав подальший розвиток в роботі [47]. Була покращена точність, збільшений ряд потенційних задач, де можна застосувати розроблений метод. Але в методу також є недоліки – метод є менш точним у випадках, коли відстані між вузлами є великими та неоднорідними.

Також варто відзначити метод скінченних точок [48], який розроблений з метою сприяння моделюванню розв’язку задач, пов’язаних зі складними геометріями, вільними поверхнями, рухомими межами та адаптивним уточненням. У подальших роботах було досягнуто конкурентоспроможного часу генерації в порівнянні з традиційними методами дискретизації, що вперше показало, що безсіткові методи є реальною альтернативою для полегшення проблем дискретизації [49].

Іншою групою безсіткових методів стали гідродинаміка згладжених частинок (англ. Smoothed Particle Hydrodynamics, SPH). Першими розробками цього методу стали роботи [65, 66]. Першою областю застосування цього методу стала гідродинаміка зірок. До переваг гідродинаміки згладжених частинок можна віднести: можливість застосування в ситуаціях з складною динамікою границь, простота розпаралелювання алгоритму, обчислювальна ефективність методу, в порівнянні з методами, що використовують сітки для дискретизації обчислювальної

Іншою групою безсіткових методів стали напівнеявні методи рухливих частинок (англ. Moving particle semi-implicit method, MPS). Першою роботою, де розроблений та застосований даний метод стала робота [50]. За допомогою даного методу були промодельовані градієнт тиску, дифузія, нестисливість, використовуючи зіткнення частинок, разом з функцією ядра. Була відмічена підвищена чисельна стабільність. Також слід відзначити ряд робіт, де були представлені модифіковані та покращені версії методу MPS - підвищення чисельної стабільності [51,52,53,54], покращення збереження імпульсу [55,56,57] поліпшення збереження механічної енергії [55], покращений розрахунок тиску [58,59,60,61], а також поліпшення моделювання багатофазних та гранульованих потоків [62,63,64].

Методи частинок також були поєднані з іншими методами Лагранжа, такими як метод дискретних елементів, щоб уможливити багатомасштабне моделювання різних фізичних явищ. Наприклад, дослідники розробили гібридні розв’язувачі MPS-DEM для імітації потоків твердої рідини, як показано в аналізі лавин уламків [67] або взаємодії потоку рідини з твердим тілом, як показано в роботі [68] на прикладі взаємодії хвилі та блоку броні перед кесонним хвилерізом. У недавньому та важливому дослідженні [69], автор об’єднав передові теорії контактної механіки з гідродинамікою згладжених частинок і представив новий підхід під назвою SPH-DCDEM (Метод розподілених контактних дискретних елементів). Цей метод забезпечує чітке та точне моделювання рідинно-твердих фаз шляхом поєднання сильних сторін SPH та методу дискретних елементів розподіленого контакту.

У контексті методів частинок, заснованих на проекції, автори роботи [70] представили реалізацію методу MPS на основі графічних процесорів. У цьому випадку прискорення було обмежено лише одним порядком величини, в основному через ітераційний процес вирішення рівняння тиску Пуассона. Такий самий порядок прискорення досягається в інших реалізаціях методів частинок на основі графічного процесора, включаючи MPS [71] та ISPH [72].

1.3 Методи моделювання на основі кліткових автоматів

На початку 1950-их років Станіслав Улам, Джон фон Нейман і Конрад Цузе представили клітинні автомати. Улам моделював розвиток дво- та тривимірних візерунків [73,74]. Джон фон Нейман запропонував самовідтворюваний клітковий автомат [75] який водночас реалізував універсальну машину Тюрінга [76]. Кожна з приблизно 200 000 клітин автомату фон Неймана містить 29 різних станів. Цузе опублікував свої ідеї щодо застосування клітинних автоматів до фізичних проблем у монографії [77]. Схожі ідеї були запропоновані чотирма роками пізніше Харді та ін. [78], у своїй роботі про комірчасті автомати з ґратковим газом HPP. Крім гідродинамічних проблем, моделі для електродинаміки і квантової теорії, в перспективі повинні були стати об’єктами досліджень Цузе. Найбільш далекосяжним баченням була його концепція Всесвіту як кліткового автомата, що охоплює гігантську кількість клітин [79].

У 1970 році Джон Хортон Конвей представив гру «Життя», двовимірний клітковий автомат з простими правилами оновлення, але складною динамікою [80]. Мартін Гарднер зробив клітинні автомати дуже популярними завдяки серії статей про «Життя» в Scientific American [80,81,82,83].

Незважаючи на прості правила оновлення, клітинні автомати мають здатність демонструвати складну поведінку, що робить їх придатними як інструменти моделювання для широкого спектру фізичних явищ, включаючи біологічні, фізичні та хімічні процеси. Кліткові автомати прості в реалізації та особливо вигідні для масивних паралельних комп’ютерів завдяки локалізованому характеру їхніх правил оновлення. Крім того, їхня характерна внутрішня структура забезпечує безумовну чисельну стабільність.

До методів автоматів граткового газу і граткового рівняння Больцмана існували подібні моделі. У 1964 році, для дослідження аеродинаміки, Бродуелл запропонував використовувати рівняння Больцмана лише з кількома дискретними швидкостями [84]. У 1973 році Харді, де Пазіс і Помо запропонували першу одношвидкісну модель кліткового автомата ґратчастого газу у двовимірному

просторі квадратної ґратки(HPP – за ініціалами трьох авторів методу) для вивчення статистичних механічних властивостей двовимірних рідин, таких як розбіжність двовимірних транспортних коефіцієнтів [85, 86]. Також цей метод має поширену в літературі назву гратчастий газовий автомат (англ. Lattice gas automata - LGA) Модель має серйозні недоліки, оскільки імпульс завжди зберігається як у горизонтальній, так і у вертикальній смугах. Жодна енергія ніколи не вилучається з моделі, ні через зіткнення, ні через рух, тому моделювання триває безскінченно довго. Модель HPP не мала обертальної інваріантності, що робило модель дуже анізотропною. Це означає, наприклад, що вихори, створені моделлю HPP, мають квадратну форму [87]. Але слід підкреслити, що і модель Бродуелла, і модель HPP були запропоновані більше як теоретичні моделі, ніж як обчислювальні інструменти.

У 1986 році Фріш, Хаслахер і Помо показали, гратковий газовий автомат з решіткою з більшою групою симетрії, ніж для квадратної ґратки, дає нестисливе рівняння Нав'є-Стокса в макроскопічній масштабі [88]. Ця модель з гексагональною симетрією названа FHP відповідно до ініціалів трьох авторів. Головним недоліком цієї моделі є складність розширення моделі для обробки тривимірних проблем, що вимагає використання більшої кількості вимірів для підтримки достатньо симетричної сітки для вирішення таких проблем [89].

Подальшого розвитку метод LGA отримав в роботі [90], в 1988 році. Щоб усунути статистичний шум, що є недоліком методу LGA, автори відмовилися від булевої операції LGA, що включає змінні зайнятості частинок, знехтувавши кореляціями частинок і ввівши усереднені функції розподілу.

Автори роботи [91] внесли значний розвиток методу LGA, запропонувавши рівняння решітки Больцмана (англ. Lattice Botzmann equation, LBE), яке включає лінеаризований оператор зіткнення. Це припущення передбачає, що розподіл близький до стану локальної рівноваги. Кілька робіт [93, 94], автори яких, незалежно один від одного, представили спрощену версію лінеаризованого оператора зіткнень на основі моделі зіткнень Бхатнагара-Гросса-Крука (англ. Bhatnagar–Gross–Krook, BGK) [92]. Решітчаста модель BGK [95, 96] використовує

функцію розподілу локальної рівноваги для відновлення макроскопічних рівнянь Нав'є-Стокса.

Загадані вище роботи вважаються основоположними в дослідженні групи методів моделювання динаміки рідин, що отримали назву решітчастою моделі Больцмана (англ. lattice Boltzmann methods, LBM) [97].

Остаточне оформлення решітчастої моделі Больцмана, як нового методу обчислювальної гідродинаміки було здійснено у роботі [112] в 1991 році. В цій роботі автори описали ідею методу, основоположні рівняння, потенційні області застосування методу, порівняння з іншими методами моделювання, перспективи майбутнього розвитку.

Згадані методи в розділах 1.1 та 1.2 зазвичай моделюють рух та динаміку рідини з нелінійних диференціальних рівнянь у часткових похідних. Ці рівняння дискретизуються та перетворюються в системи лінійних алгебраїчних рівнянь, що розв'язуються тими чи іншими чисельними методами. Узагальнено такий підхід називається методом «зверху-вниз». Методи LGA та LBM, є варіантами методу «знизу-вгору» де початковою точкою є дискретна мікроскопічна модель, яка через власну структуру зберігає необхідні величини (масу та імпульс для рівняння Нав'є-Стокса). Схематичне порівняння цих типів методів зображене на рис. 1.2

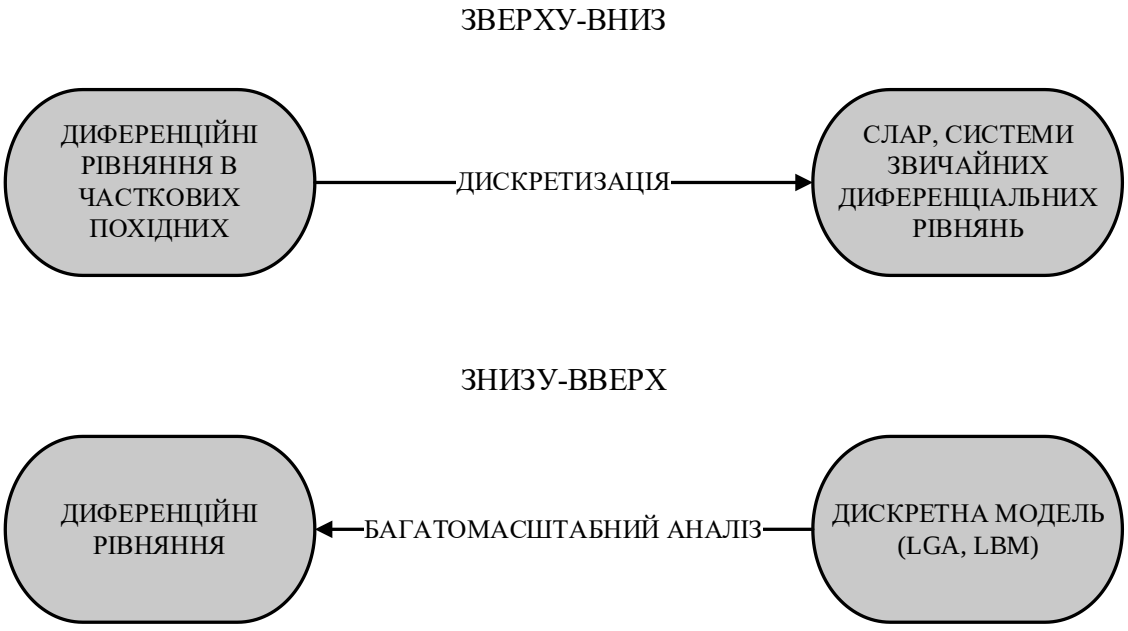

Рисунок 1.2 – Порівняння методів «зверху-вниз» та «знизу-вверх»

З розвитком методів на основі решітчастої моделі Больцмана, постала проблема дослідження та розрахунку різних типів граничних умов. Одним із перших способів моделювання нековзкої суцільної стінки, як граничної умови, є умова зворотного відображення (англ. bounce back rule) [98, 99]. Проте, подальші роботи показали недостатню точність такого підходу [100,101,102]. Після цього було вкладено значні зусилля в розробку більш точних схем граничних умов, ніж проста умова зворотного відображення. Спочатку обробка граничних умов виникла як різні умови зворотного відображення, але їхнє застосування було обмежено прямими та регулярними геометріями границь, такими як плоскі стіни [103,104]. Щоб застосувати граничну схему, необхідно мати інформацію про швидкість та/або щільність у відповідному вузлі. Крім того, ці методи певною мірою залежать від певних властивостей потоку. Прикладом цього є гідродинамічна гранична умова [105], де фіксована внутрішня енергія встановлюється рівною квадрату швидкості звуку для стаціонарного гідродинамічного поля.

Згодом було докладено більше зусиль для розробки методів, здатних працювати з криволінійними межами. У 1998 році був запропонований локалізований підхід під назвою FH у кількох роботах[106,107]. Однак цей алгоритм стає нестабільним, коли параметр часової релаксації τ наближається до 1, оскільки він включає фактор $1/(\tau - 1)$. Автори роботи [108,109] запропонували вдосконалення, відоме як MLS, яке включає додатковий сусідній вузол для розширення області стабільності. Однак це вдосконалення не усуває властивий FH недолік. У 2001 році у роботі [110] був представлений інший метод, заснований на зв’язку, під назвою BFL, який використовує лінійну інтерполяцію з одним або двома сусідніми вузлами. Пізніше було представлене більш загальне рішення в роботі[111]. Автори роботи [113] запропонували уніфіковану схему обробки криволінійних стін другого порядку точності. Методи, згадані тут, можна розглядати як пряме розширення умови зворотного відображення.

Окремої уваги заслуговують ряд робіт, де було досліджено моделі решіток, що були застосовані для дискретизації рівняння Больцмана в методі LBM. Однією із перший таких робіт стала робота [114] 1993 року, де були представлені

потенційні двовимірні моделі решіток для побудови схем. В цьому дослідженні розглядалися квадратна модель з дев'ятьма напрямками руху частинок та шестикутна модель з сімома напрямками. Робота також описувала спосіб задання в'язкості рідини за допомогою параметра релаксації і методи встановлення початкових і граничних умов. Того ж року автори роботи [115] провели порівняльний аналіз між LBM та спектральним методом. Порівняння показало високу точність отриманих розв'язків і зауважено, що LBM є в 2,5 рази

В’язкий потік у квадратній порожнині для широкого діапазону чисел Рейнольдса, був детально досліджений в роботі [116], використовуючи модель LBM-SRT із граничною умовою відскоку. В цій роботі була використана двовимірна дев'ятишвидкісна модель для дослідження зв'язку між роміром комірки та точність моделювання. Моделювання потоку рідини в порожнині є досить популярною в дослідників задачею та моделю для тестування нових методів моделювання [117,118,119].

Дослідження різноманітних дво- та тривимірних моделей решіток та подальше вдосконалення методу LBM було зроблене в 1997 році [120]. Так були досліджені двовимірні схеми з шістьома, сімома та дев’ятьма швидкостями, а також тривимірна схема з двадцяти сімома дискретними швидкостями. Того ж року була опублікована велика робота, де були розглянуті методи задання граничних умов для широко типу прикладних проблем [121]. Автори [122] в своїй роботі підтвердили можливість виведення основних рівнянь гідродинаміки з решітчастого рівняння Больцмана. На прикладі моделювання розв’язання проблеми течії рідини в плоскій порожнині, були досліджені стійкість та границі застосування чисельних схем.

Були опубліковані ряд робіт, де методи LBM порівнювались з іншими традиційними методами моделювання руху рідин – з методом скінченних об’ємів [123], скінченних різниць [124], скінченних елементів [125], зі спектральними методами [126]. Всі згадані дослідження підтвердили перевагу методів LBM.

Також відзначена ефективність решітчастої моделі Больцмана в моделюванні складних геометрій, де стінки, що виступають в ролі границь, надзвичайно складні.

Так, в 1997 році була опублікована робота, де був змодельований рух рідини навколо циліндру [127]. Також автори розрахували силу, яку здійснює потім на тверде тіло під час свого руху. Схожі дослідження були здійснені в наступній роботі [128], де автори змогли охарактеризувати рух рідини навколо циліндру, ввівши коефіцієнти тяги і підняття, у вигляді формул.

Перше застосування решітчастої моделі Больцмана для моделювання багатокомпонентної рідини було описане в роботі [129]. На прикладі процесу конденсації рідких крапель у перенасиченому розчині, були розглянуті способи задання та обрахунку граничних умов, основні рівняння та чисельні схеми.

Нелінійний багатосітковий підхід для вирішення дискретного рівняння Больцмана був вперше запропонований в роботі [130]. У статті [131] представлені ефективні стратегії розв’язання стаціонарного гратчастого рівняння Больцмана та дослідив застосовність багатосіткового LBM до проблеми руху потоку рідини в порожнині (англ. driven cavity). Незважаючи на те, що багатосіточні методи є перспективними для LBM, подібно до традиційних методів обчислювальної гідродинаміки, проблеми в етапах подовження та обмеження поблизу стінок складної форми не були достатньо досліджені. Для підвищення чисельної точності в LBM були введені нерівномірні сітки. Автори роботи [132] описали решітчасту модель Больцмана для моделювання рівняння Нав’є-Стокса на довільних нерівномірних сітках. Вони представили результати течії в двовимірному симетричному каналі з раптовим розширенням. У роботі [133] автори використали метод Больцмана з подвійною заселеністю решітки з нерівномірною сіткою для імітації природної конвекції в квадратній порожнині. Щоб зберегти властиві переваги LBM, такі як простота кодування та ефективність обчислень, перевага надалась використанню однорідної решітки.

У галузі обчислювальної гідродинаміки спостерігається значний прогрес у числових розв’язках рівнянь Ейлера. У минулому звичайний меод LBM зіткнувся з обмеженнями в при моделюванні руху рідин з високими числами Маха через дискретний характер швидкостей частинок. Щоб вирішити це, у рамках LBM необхідно вирішити дві ключові проблеми. По-перше, розширити допустимий

діапазон чисел Маха, а по-друге, необхідно ввести зміну температури в структуру моделей LBM.

Стисливі рівняння Нав'є-Стокса спочатку були введені в рамках LBM в роботі [134]. Автори запропонували вибіркову модель швидкості звуку, яка ефективно імітує стисливий потік шляхом відповідного налаштування параметрів рівноважної функції розподілу для мінімізації швидкості звуку. У роботі [135] автори розробили решітчасту модель Больцмана без членів нелінійного відхилення. Двовимірна дев'ятибітна модель з двома рівнями енергії спокою, яка точно відновлює рівняння Ейлера через процеси потоку та зіткнення, була розроблена авторами роботи [136]. Ця модель успішно змоделювала проблему ударної труби Сода та Лакса. Автори роботи [137] запровадили силу притягання, щоб зменшити швидкість звуку та пом'якшити обмеження, накладені низькими числами Маха. Вони продемонстрували моделювання потоків стисливої рідини з числами Маха до 5.

Для моделювання коливань температури в стисливих течіях, була сформульована теплова модель в роботі [138]. У роботі [139] автори запропонували гратчасту модель Больцмана для стисливих рівнянь Нав'є-Стокса з гнучким відношенням питомої теплоємності. Автори роботи [140] представили модель Больцмана стисливої гратки та успішно застосував її для порівняння задач стисливого потоку в двовимірному та тривимірному просторах. Модель функції подвійного розподілу, що дискретизує керівні рівняння, використовуючи монотонну схему третього порядку для скалярної схеми законів збереження кінцевого об'єму була показана в роботі [141].

З розвитком, збільшенням потужності графічних прискорювачів в 2000-х роках, і разом із появою спеціалізованого програмного забезпечення, що дозволило здійснювати обчислення, за допомогою графічних прискорювачів, було помічено появо цілого ряду робіт, присвяченого ефективності використання графічних прискорювачів для моделювання руху рідин методами LBM.

Реалізації методів LBM на графічних проскорювачах почали розроблятись з початку 2000-х років. Так однією із перших робіт стала стаття [142], де була

продемонстрована економічна ефективність реалізації. З тих пір багато робіт [143, 144, 145], запропонували методи оптимізації LBM на графічних проскорювачах. Основні стратегії стосуються розташування пам’яті, оскільки продуктивність здебільшого обмежена пропускною здатністю пам’яті. Випуск програмного забезпечення CUDA компанією Nvidia [146] сприяв розробці спеціальних алгоритмів для графічних прискорювачів, і воно використовується для реалізації методів LBM на графічних прискорювачах в переважній більшості робіт.

Робота корейських дослідників показала очевидну перевагу у використанні графічних прискорювачів над звичайними процесорами при моделюванні руху потоків рідин у тривимірному просторі, методом LBM. Так графічний прискорювач Nvidia Tesla K20 GPU показав в 136 швидшу швидкість обчислень, в порівнянні з процесором 6-core 2.2 Ghz Intel Xeon [147].

Визнання того, що LBM має можливість моделювати розв’язки диференціальних рівнянь з частковими похідними, охоплюючи системи, які виходять за межі сфери фізики, що лежить в основі статистичної фізики та кінетичної теорії, було визнано протягом досить тривалого часу. Однак всеосяжна теорія, яка б конкретно розглядала "цифрові LBM", досі відсутня.

Приклади застосування методів LBM, поза межами обчислювальної гідродинаміки, наступні:

- LBM для рівнянь Максвелла для електродинаміки [148, 149], з наголосом на поширенні електромагнітних хвиль, з розширенням на магнітогідродинаміку [150,151]
- релятивістські рідини, для застосування в астрофізиці [152]
- передача тепла, за допомогою рівняння провідності та переносу випромінювання [153]
- моделювання епідемій[154] та динаміки натовпу[155]

Підсумувавши зроблений огляд методу LBM, можна виділити його основні переваги в моделюванні руху рідин:

1. Метод базується на моделюванні руху індивідуальних частинок, що спрощує обчислення.

2. Метод LBM добре підходить для паралельного обчислення, що дозволяє розподілити обчислювальне навантаження між багатьма процесорами або обчислювальними вузлами та дозволяє прискорити час розрахунків та моделювання більш складних систем.

3. Метод LBM може легко моделювати складні геометрії, включаючи пористі матеріали або нерегулярні форми, завдяки використанню сітки, яка відповідає фізичній геометрії системи.

Але також слід згадати і основні недоліки цього методу:

1. Обмеження на низькі числа Маха
2. Точність та складність моделювання турбулентного потоку
3. Труднощі при моделюванні нестисливих рідин

Ці недоліки створюють потребу на нові методи та підходи, що дозволять розширити сферу застосування методів LBM, зберігаючи властиві їм переваги.

1.4 Методи машинного навчання

Машинне навчання — це підгалузь штучного інтелекту, яка зосереджується на розробці алгоритмів і моделей, які дозволяють комп’ютерам навчатися та робити прогнози чи рішення без явного програмування. Це пов’язано з проектуванням і створенням систем, які можуть вчитися та адаптуватися до даних, покращуючи їх продуктивність з часом.

У традиційному програмуванні розробники пишуть конкретні інструкції, яких комп’ютер повинен виконувати. Однак у машинному навчанні замість явних правил програмування алгоритми навчаються на даних, щоб вивчати шаблони, зв’язки та правила, властиві даним. Потім ці алгоритми роблять прогнози або вживають дій на основі вивчених закономірностей.

В останні роки зацікавленість у застосуванні машинного навчання для розв’язання задач обчислювальної гідродинаміки значно зросла.

Однією із перших робіт в області застосування машинного навчання в обчислювальній гідродинаміці стала робота 1984 року [156], де автори оцінили як застосування машинного навчання може прискорити здобуття нових знань, як

традиційні обчислювальні методи можуть використати переваги машинного навчання. Також були розглянуті вимоги до обчислювальних ресурсів, які можуть бути необхідні для використання засобів машинного навчання і штучного інтелекту. Однак останній аспект на даний час є не актуальним, оскільки обчислювальні потужності значно зросли в порівнянні з моментом написання статті. В 1980-х роках з'явився ряд робіт, що встановили зв'язки між алгоритмами машинного навчання та статистичною механікою – це були статті [157, 158, 159].

В 1990-х роках вийшов ряд статей, предметом досліджень яких стало застосування штучних нейронних мереж для задач пов'язаних з рухом потоків рідин. Автори роботи [160] використали нейронні мережі для виміру швидкості частинок, що були сфотографовані в потоці рідини (англ. Particle Image Velocimetry, PIV). Подібне дослідження було проведене у роботі [161], де були порівняні різні архітектури нейронних мереж, що були використані для виміру швидкості та структури потоку рідини.

Багатошаровий персептрон для вирішення нелінійних задач, таких як рівняння Пуассона і теплопровідність з нелінійним виділенням тепла, був розроблений у роботі [162]. Основним недоліком цього методу є відносно малий розмір модельованої області [163]. Крім того, автори не згадали швидкість роботи свого методу. Автори статті [164] запропонували методологію розробки нейронних алгоритмів для моделювання розв'язку диференціальних рівнянь. Основними недоліками досліджень зазначеного періоду є вузький набір граничних умов, специфічний діапазон модельованих функцій вільного члена та малі розміри областей моделювання. Недоліками двох останніх робіт є відсутність згадки про швидкісні властивості методів.

Початок 2000-х років продемонстрував значне підвищення обчислювальної ефективності. У результаті було опубліковано набагато більше досліджень, які представили більш складні та надійні моделі. Було запропоновано багатошарові персептрони для прогнозування ортогонального розкладання 2D рівняння Нав'є-Стокса та 1D рівняння Курамото-Сивашинського [165]. Попередній метод можна

адаптувати до випадку моделювання розв’язку рівняння Пуассона, запропонований метод не обробляє граничні умови Неймана.

В останні роки, однією із цілей досліджень застосування машинного навчання стало прискорення традиційних методів чисельних моделювань. Були запропоновані методи на основі глибоких нейронних мереж, що покращили схеми дискретизації. Наприклад, в роботі [166] автори запропонували метод для оцінки просторових похідних у сітках з низькою роздільною здатністю, перевершуючи методи скінченних різниць. Недоліками такого підходу є незадовільна швидкість обчислень та складність адаптації підходу для складних обчислювальних областей. Автори роботи [167] розробили подібний підхід, але з іншою метою – для підвищення точності скінченно різницевих схем п’ятого порядку. Однак основним недоліком цієї роботи є застосування методу лише для одновимірного простору.

У роботі [168] автори представили використання глибокої нейронної мережі для реалізації широко використовуваної схеми дискретизації кінцевого об’єму для моделювання руху рідини. Автори провели експерименти з використанням реактивних потоків, які показали задовільну точність, в порівнянні з еталонними даними високої роздільної здатності, в той же час показавши збільшення обчислювальної швидкості в 10 разів. Але метод має значний недолік – зі зростанням часу моделювання, точність методу падала.

Для підвищення точності методів скінченних різниць/скінченних об’ємів була запропонована нейронна мережа на основі згорткових шарів та шарів довго-короткострокової пам’яті (англ. long-short-term-memory, LSTM) [169]. Даний метод показав значну точність, але в роботі не згадуються показники обчислювальної швидкості.

Авторами статті. [170] був розроблений метод корекції між областями моделювання низького та високого розширення, для потоку Колмогорова [171] в двовимірному просторі. Перевагою розробленого методу є більша точність під час довгих симуляцій.

Окремо варто згадати праці, де була досліджена можливість застосування алгоритмів машинного навчання для моделювання розв’язку рівняння Пуассона,

яке є ключовою складовою методів обчислювальної гідродинаміки, які базуються на розв'язанні рівняння Нав'є-Стокса, а також використовується в електродинаміці [172,173]. Дослідники. в своїй роботі [174] розробила структуру нейронної мережі для розв’язання рівняння Пуассона під час моделювання руху нестисливих рідин. До переваг цього методу можна віднести набагато кращу швидкодію, в порівнянні з традиційним методом Якобі для вирішення СЛАР, та задовільну точність методу при низьких значеннях числа Річардсона. Проте, при більших значеннях числа Річардсона, точність була гіршою.

Значне прискорення в моделювання розв’язку рівняння Пуассона було продемонстроване в роботі [175], де застосування згорткової нейронної мережі стало одним із кроків в багатосітковому методі, що був застосований при моделюванні нестисливих рідин.

Автори роботи [176] розробили структуру нейронної мережі, що дозволила вирішити рівняння Пуассона шляхом поділу задачі на гомогенну задачу Пуассона та чотири негомогенні підзадачі Лапласа. Також була продемонстрована задовільна точність розв’язку, що відкриває можливість для використання даного алгоритму як початкове припущення при застосуванні ітераційних алгоритмів.

Іншим типом задач, де застосування машинного навчання почало досліджуватись тільки в останні 15 років стали RANS-моделі (англ. Reynolds-Averaged Navier-Stokes), оскільки що моделювання RANS є ненадійними для потоків із сильним градієнтом тиску та кривизною [177]. Обмежена точність передбачення залежних від режиму, насичених фізикою явищ у турбулентних потоках пояснюється недійсністю певних припущень у моделі закриття, таких як припущення про вихрову в’язкість. Ці невизначеності, що виникають через традиційне замикання напруги Рейнольдса, вносять значні розбіжності у форму моделі, які перешкоджають передбачуваності моделей RANS, створюючи постійні проблеми в моделюванні RANS. Однією із перших робіт в цій області стала робота [178], де автори за допомогою глибоких нейронних мереж передбачили повну розбіжність у турбулентній в’язкості, використовуючи дані, отримані з методів прямої чисельної симуляції, про потік у плоскому каналі, а потім вони передбачили

потоки в каналах із хвилястими межами. Продовження дослідження в цій області отримали в подальших роботах [179, 180, 181]. Однак внутрішнім обмеженням цих підходів є те, що повна інверсія поля виконується в просторі фізичних координат, що обмежує подальше застосування прогнозів для потоків з різними конфігураціями. Подолання цієї проблеми за допомогою машинного навчання стало предметом досліджень подальших робіт [182,183].

Також машинне навчання стало популярним засобом для поліпшень та модифікації існуючих методів LBM, застосування яких було описане в розділі 1.3. В роботі [184] автори запропонували нейронну мережу, що моделює нерівномірний сталий ламінарний потік у дво- та тривимірній області, із заданими граничними умовами. Він складається з трьох ключових компонентів: адаптованої функції відстані, як гнучкого та загального геометричного представлення для згорткових нейронних мереж; кілька згорткових шарів для кодування для вилучення абстрактних і високорівневих геометричних зображень; кілька згорткових шарів для декодування, які відображають абстрактні геометричні представлення в обчислювальному полі швидкості гідродинаміки. Основна перевага цього підходу полягає в тому, що прогноз CNN нерівномірного сталого ламінарного потоку є значно швидшим, ніж традиційні методи LBM. Також варто відзначити метод підбору параметрів оператора зіткнень MRT за допомогою глибокої нейронної мережі [185].

В останні роки популярним об’єктом досліджень стали фізико-інформовані нейронні мережі (англ. Physics-informed neural networks, PINN)) [226]. фізико-інформовані нейронні мережі — це клас моделей машинного навчання, які використовуються для моделювання розв’язання та наближення рівнянь у частинних похідних та інших фізичних законів, які керують природними явищами. Ці нейронні мережі розроблено для включення законів фізики безпосередньо у свою архітектуру, що робить їх особливо корисними для моделювання та імітації фізичних систем. Це дозволяє регуляризувати процес навчання навчання нейронної мережі, забезпечивши більшу точність роботи моделі. Серед досліджень в цій області варто відзначити роботу [227] присвячену застосування PINN для

моделювання нестисливих рідин на основі рівнянь Нав’є-Стокса. Також даний підхід був використаний для моделювання руху рідин в біологічних об’єктах [228]. Зі збільшенням числа досліджень в області фізико-інформованих нейронних мереж, були створені програмні засоби для дослідження та впровадження моделей, наприклад, Modulus від компанії NVIDIA [229]. Недоліками даного типу нейронних мереж є складнощі при розв’язанні певних типі диференційних рівнянь та знаходження глобального оптимуму в процесі навчання моделей [230, 231].

Узагальнивши огляд досліджень застосування машинного навчання в моделюванні динаміки руху рідин, можна виділити такі основні переваги:

- обчислювальна ефективність та швидкість
- адаптивність
- спроможність використовувати складні структури даних

Недоліки використання машинного навчання наступні:

- наявність даних для навчання та їх якість
- можливість інтерпретації моделі
- перенавчання

1.5 Висновки до розділу 1

У цьому розділі було проведено огляд методів моделювання динаміки рідини. В результаті літературного аналізу було виявлено і досліджено чотири основних підходи до моделювання руху рідини: методи моделювання на основі рівняння Нав'є-Стокса, методи моделювання розв’язку рівняння Пуассона, методи LBM та методи машинного навчання.

Перший підхід, заснований на рівнянні Нав'є-Стокса, є широко використовуваним у гідродинамічних дослідженнях. Він базується на наближеннях та уточненнях рівнянь Нав'є-Стокса, що дозволяють враховувати різноманітні фізичні властивості рідини та її руху. Цей підхід дозволяє моделювати різноманітні гідродинамічні явища та процеси, такі як течія рідини, турбулентні потоки та інші.

Другий підхід, що використовує методи розв'язання рівняння Пуассона, зазвичай застосовується для вирішення проблем гідродинаміки рідини. Ці методи засновані на числових методах розв'язання рівняння Пуассона, що дозволяють встановлювати розподіл тиску в більшості методах обчислювальної гідродинаміки.

Третій підхід, методи LBM, представляють собою альтернативний підхід до моделювання динаміки рідини. Вони базуються на статистичних методах, де рух рідини моделюється через мікроскопічні елементи, які знаходяться на вузлових точках регулярної сітки. Ці методи володіють деякими перевагами, такими як простота реалізації та здатність до паралельного обчислення.

Четвертий підхід - це методи машинного навчання. Вони стали все більш популярними у сучасному моделюванні динаміки рідин. Ці методи використовуються для аналізу великого обсягу даних, отриманих з експериментів або симуляцій, з метою побудови моделей, передбачення та оптимізації різних гідродинамічних процесів. Вони можуть забезпечити нові можливості для вирішення складних проблем, але вимагають великого обсягу даних та обчислювальних ресурсів для своєї реалізації.

В цьому розділі було проаналізовано основні переваги, обмеження та застосування кожного з цих методів моделювання динаміки рідини. Враховуючи всі переваги та недоліки кожного типу методів, доцільним є розробка дворівневого методу моделювання руху потоку рідини, за допомогою методу LBM та машинного навчання, з метою нівелювання недоліків цих методів, використавши їх основні переваги.

РОЗДІЛ 2
МОДИФІКОВАНИЙ МЕТОД LBM

2.1 Решітчаста модель Больцмана

Метод решітчастої моделі Больцмана (англ. Lattice Boltzmann Method, LBM), є числовим методом для моделювання газу або рідини на мікроскопічному рівні. Цей метод базується на кінетичному рівнянні Больцмана і широко використовується в комп'ютерній гідродинаміці та комп'ютерній фізиці рідин.

Основна ідея методу LBM полягає в тому, що рідина або газ моделюються як сукупність "частинок" або "вузлів", розташованих на однорідній решітці. Кожна частинка має певний набір властивостей, таких як швидкість і щільність. Взаємодія між частинками відбувається шляхом обміну кількості руху і енергії на основі виконання простих макроскопічних правил.

2.1.1 Метод LBM в контексті обчислювальної гідродинаміки

Для переважної більшості методів обчислювальної гідродинаміки достатньо описувати стан рідини за допомогою набору макроскопічних змінних, таких як тиск і швидкість, які виникають у результаті статистичного усереднення динаміки мікроскопічних частинок. У цьому випадку рідину можна розглядати як суцільне середовище, а головні керівні рівняння – так рівняння Нав’є-Стокса – спираються на деякі глобальні закони збереження. Однак добре відомо, що гіпотеза континууму становить найвищий рівень абстракції для рідини, повністю ігноруючи мікроскопічний світ, що лежить в основі. Відтворення детальної динаміки на мікроскопічному рівні, з іншого боку, вимагає значних обчислювальних ресурсів. Метод LBM знаходиться між мікроскопічним та макроскопічним рівнями абстракції, на мезоскопічному рівні, яке описує суцільне середовище через усереднені параметри деякого числа частинок (рис 2.1).

Оскільки основою методу LBM є кінетичне рівняння Больцмана, що описує статистичний розподіл частинок в газі чи рідині, в подальших частинах розділу покажемо зв’язок між гідродинамікою та кінематикою частинок.

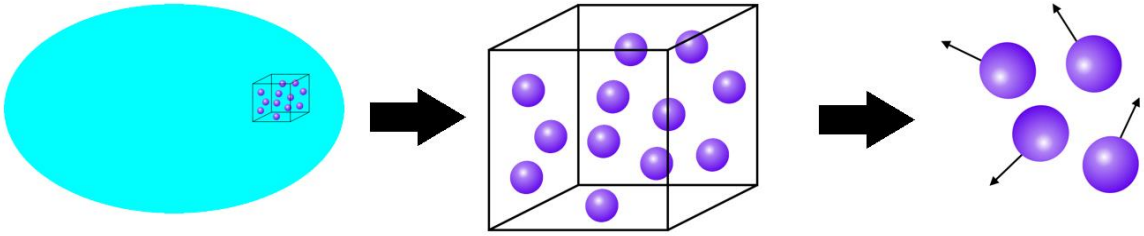

Рисунок 2.1 Рівні абстракції опису суцільного середовища, зліва направо: макроскопічний рівень, мезоскопічний рівень, мікроскопічний рівень

2.1.2 Функція розподілу

Розглянемо газ, що складається з N частинок, розташованих у кубічному просторі об'ємом V. Ми припускаємо, що газ є розрідженим, тому ми можемо зосередитись лише на подвійних зіткненнях частинок, і ігноруємо можливість, три чи або більше частинок зіштовхнутися одночасно. Додатково, ми припускаємо, що газ є ідеальним, що означає, що він підпорядковується рівнянню стану ідеального газу:

$$p = \rho c_s^2 \tag{2.1}$$

де p — тиск газу, ρ — густина маси, c_s — теплова швидкість або швидкість звуку, визначена як

$$c_s = \sqrt{\frac{k_B T}{m}} \tag{2.2}$$

де k_B — постійна Больцмана, T — температура газу, m — маса частинки.

Припустимо, що газ однорідний, де кожна частинка має однакову масу m, має положення в тривимірному просторі $\boldsymbol{x} = (x, y, z)$ та швидкість $\boldsymbol{v} = (v_x, v_y, v_z)$. Таким чином стан кожної частинки в певний момент часу t описати як точку в шестивимірному просторі [186].

Введемо функцію розподілу для згаданих вище *N* точок в момент часу *t*. Ця функція має вигляд $f(\boldsymbol{x}, \boldsymbol{v}, t)$. Тоді кількість частинок в момент часу *t*, що знаходяться в інтервалі $\boldsymbol{x} + d\boldsymbol{x}$, а їхні швидкості знаходяться в інтервалі $\boldsymbol{v} + d\boldsymbol{v}$ дорівнює:

$$dN = f(\mathbf{x}, \boldsymbol{v}, t)\mathrm{d}\boldsymbol{x}d\boldsymbol{v} \tag{2.3}$$

Тоді повна кількість частинок *N* в об'ємі *V* дорівнює:

$$\int f(\boldsymbol{x}, \boldsymbol{v}, t)d\boldsymbol{x}d\boldsymbol{v} = N \tag{2.4}$$

Щоб функція розподілу частинок могла вважатись неперервною, значення $d\boldsymbol{x}d\boldsymbol{v}$ повинно бути достатньо малим в порівнянні з розмірами досліджуваної області, і в той же час достатньо великим, щоб містити значну кількість частинок.

Властивості газу в певний момент часу *t* і точці простору **x** можна вивести із функції розподілу:

$$\rho(\boldsymbol{x}, t) = m\int f d\boldsymbol{v} \tag{2.5}$$

де $\rho(\boldsymbol{x}, t)$ – густина.

2.1.3 Рівняння Больцмана

Розглянемо розріджений газ, в якому відбуваються лише парні зіткнення. Після зіткнення швидкість частинок змінюється, внаслідок цього відбувається переміщення точок в описаному вище шестивимірному просторі $(\boldsymbol{x}, \boldsymbol{v})$. Таку зміну кількості частинок можна зобразити у вигляді наступного рівняння:

$$\frac{d(dN)}{dt} = (N_{in} - N_{out})d\boldsymbol{x}d\boldsymbol{v} \tag{2.6}$$

де N_{out} – кількість частинок, що залишили об'єм $d\boldsymbol{x}d\boldsymbol{v}$, N_{in} – кількість частинок, що увійшла відповідно.

Використавши формулу (2.3), рівняння (2.6) можна зобразити наступним чином:

$$\frac{d(dN)}{dt} = \frac{f(\boldsymbol{x}, \boldsymbol{v}, t)\mathrm{d}\boldsymbol{x}d\boldsymbol{v}}{dt} = (N_{in} - N_{out})d\boldsymbol{x}d\boldsymbol{v} \tag{2.7}$$

Поділивши рівняння (2.7) на $d\boldsymbol{x}d\boldsymbol{v}$, отримуємо:

$$\frac{f(\boldsymbol{x},\boldsymbol{v},t)}{dt} = N_{in} - N_{out} \tag{2.8}$$

де величина $N_{coll} = N_{in} - N_{out}$ називається інтегралом зіткнень [187].

Продиференціюємо функцію розподілу:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial z}\frac{dz}{dt} + \frac{\partial f}{\partial v_x}\frac{dv_x}{dt} + \frac{\partial f}{\partial v_y}\frac{dv_y}{dt} + \frac{\partial f}{\partial v_z}\frac{dv_z}{dt} = \\ = \frac{\partial f}{\partial t} + \frac{d\mathbf{x}}{dt}\frac{\partial f}{\partial \boldsymbol{x}} + \frac{d\boldsymbol{v}}{dt}\frac{\partial f}{\partial \boldsymbol{v}} \tag{2.9}$$

Оскільки $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}, \frac{d\boldsymbol{v}}{dt} = \frac{\boldsymbol{F}}{m}$, то рівняння (2.9) набуває вигляду:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \boldsymbol{v}\frac{\partial f}{\partial \boldsymbol{x}} + \frac{\mathbf{F}}{m}\frac{\partial f}{\partial \boldsymbol{v}} \tag{2.10}$$

Прирівнявши праву частину рівняння (2.10) та ліву частину рівняння (2.8) маємо:

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\frac{\partial f}{\partial \boldsymbol{x}} + \frac{\boldsymbol{F}}{m}\frac{\partial f}{\partial \boldsymbol{v}} = N_{\text{coll}} \tag{2.11}$$

Що є кінетичним рівнянням Больцмана [188]

2.1.4 Зв'язок кінетичної теорії газів та гідродинаміки

Розглянемо газ, що має середню швидкість $\boldsymbol{u}$ в точці $\boldsymbol{x}$ і в момент часу *t*. Кожна частинка газу має швидкість $\boldsymbol{v}$, що складається зі швидкості $\boldsymbol{u}$ та швидкості $\boldsymbol{v_c}$ – випадкового відхилення [189]. Швидкість $\boldsymbol{v_c}$ – називають тепловою або власною швидкістю.

Введемо умовну функцію швидкості власного руху $g(\boldsymbol{v_c})$. Середнє значення довільної функції *F* дорівнює:

$$\langle F \rangle = \frac{\int F f d\boldsymbol{v}}{\int f d\boldsymbol{v}} \tag{2.12}$$

Тоді середнє значення функції $g(\boldsymbol{v_c})$ дорівнює:

$$\langle g \rangle = \frac{\int g f d\boldsymbol{v}}{\int f d\boldsymbol{v}} = \frac{1}{N}\int g(\boldsymbol{v_c}) f d\boldsymbol{v} \tag{2.13}$$

Перетворимо рівняння (2.11), помноживши його на функцію швидкості власного руху та візьмем інтеграл по швидкостям:

$$\int \frac{\partial f}{\partial t} g d\boldsymbol{v} + \int \boldsymbol{v} \frac{\partial f}{\partial \boldsymbol{x}} g d\boldsymbol{v} + \int \frac{\boldsymbol{F}}{m} \frac{\partial f}{\partial \boldsymbol{v}} g d\boldsymbol{v} = \int N_{coll} g d\boldsymbol{v} \tag{2.14}$$

Використовуючи формулу (2.13), змінимо ліву частину рівняння:

$$\begin{aligned} &\int \frac{\partial f}{\partial t} g d\boldsymbol{v} = \frac{\partial}{\partial t} \int f g d\boldsymbol{v} = \frac{\partial}{\partial t} (\langle g \rangle N) = \langle g \rangle \frac{\partial N}{\partial t} + N \frac{\partial \langle g \rangle}{\partial t} \\ &\int \boldsymbol{v} \frac{\partial f}{\partial \boldsymbol{x}} g d\boldsymbol{v} = \frac{\partial}{\partial \boldsymbol{x}} \int g \boldsymbol{v} f d\boldsymbol{v} = \frac{\partial}{\partial \boldsymbol{x}} (N \langle g \boldsymbol{v} \rangle) \\ &\int \frac{\boldsymbol{F}}{m} \frac{\partial f}{\partial \boldsymbol{v}} g d\boldsymbol{v} = \frac{\boldsymbol{F}}{m} \int \frac{\partial f}{\partial \boldsymbol{v}} g d\boldsymbol{v} = \frac{\boldsymbol{F}}{m} \left[\int \frac{\partial (g f)}{\partial \boldsymbol{v}} d\boldsymbol{v} - \int f \frac{\partial g}{\partial \boldsymbol{v}} d\boldsymbol{v} \right] = -N \frac{\boldsymbol{F}}{m} \left\langle \frac{\partial g}{\partial \boldsymbol{v}} \right\rangle \end{aligned} \tag{2.15}$$

Поєднавши рівняння (2.14) та (2.15) маємо:

$$\begin{aligned} &\langle g \rangle \frac{\partial N}{\partial t} + N \frac{\partial \langle g \rangle}{\partial t} + \frac{\partial}{\partial \boldsymbol{x}} (N \langle g \boldsymbol{v} \rangle) - N \frac{\boldsymbol{F}}{m} \left\langle \frac{\partial g}{\partial \boldsymbol{v}} \right\rangle = \int N_{coll} g d\boldsymbol{v} \\ &\frac{\partial (N \langle g \rangle)}{\partial t} + \frac{\partial}{\partial \boldsymbol{x}} (N \langle g \boldsymbol{v} \rangle) - N \frac{\boldsymbol{F}}{m} \left\langle \frac{\partial g}{\partial \boldsymbol{v}} \right\rangle = \int N_{coll} g d\boldsymbol{v} \end{aligned} \tag{2.16}$$

Таким чином ми отримали узагальнене рівняння переносу або рівняння Енського [190]. З врахуванням того, що $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{v_c}$, та макроскопічної густини дорівнює $\rho = mN$, то для $g = m$, рівняння переносу має вигляд:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho \boldsymbol{u})}{\partial \boldsymbol{x}} = 0 \tag{2.17}$$

тобто рівнянням нерозривності[191], яке можна застосувати також до рідин.

Таким чином ми показали зв'язок кінетичної теорії та гідродинаміки.

2.1.4 Дискретне рівняння Больцмана

Головною задачею решітчастої моделі Больцмана є розв'язання решітчастого рівняння Больцмана, дискретної версії рівняння (2.11), що дискретизовано в шестивимірному просторі $(\boldsymbol{x}, \boldsymbol{v})$ (розділ 2.1.2).

Припускаємо, що зовнішні сили відсутні $\boldsymbol{F} = 0$. Внаслідок цього рівняння Больцмана буде мати вигляд:

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \frac{\partial f}{\partial \boldsymbol{x}} = N_{\text{coll}} \tag{2.18}$$

Як було вже сказано, функція розподілу f залежить від простору, часу та швидкості. Спочатку проводимо дискретизацію по швидкостям. Для вводимо скінчену множину швидкостей $\boldsymbol{v_i}$, де для кожної швидкості з цієї множини вводиться дискретне рівняння Больцмана $f_i(\boldsymbol{x}, t)$. Таким чином рівняння (2.18) набуває вигляду:

$$\frac{\partial f_i(\boldsymbol{x}, t)}{\partial t} + \boldsymbol{v_i}\frac{\partial f_i(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} = (N_{\text{coll}})_i \tag{2.19}$$

де $(N_{\text{coll}})_i$ – оператор зіткнень, що представляє швидкість зміни розподілу частинок через зіткнення для швидкості $\boldsymbol{v_i}$.

Далі проводимо дискретизацію за простором і часом, та зробимо рівняння безрозмірним, ввівши для цього допоміжні змінні: L – масштаб довжини, U – контрольна швидкість, n_r – контрольна густина, t_c – час між зіткненнями. Тоді рівняння (2.19) можна представити у вигляді:

$$\frac{\partial F_i(\boldsymbol{x}, t)}{\partial \hat{t}} + L\boldsymbol{c_i}\frac{\partial F_i(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} = \frac{L}{U}(N_{\text{coll}})_i \tag{2.20}$$

де $\boldsymbol{c_i} = \boldsymbol{v_i}/U, \hat{\nabla} = L\nabla, \hat{t} = t \cdot U/L, F_i = f_i/n_r$.

Після даних перетворень, дискретизоване рівняння (2.20) буде наступним:

$$\begin{aligned} &\frac{F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t})}{\Delta\hat{t}} + c_{ix}\frac{F_i(\hat{\boldsymbol{x}} + \Delta\hat{x}, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t})}{\Delta\hat{x}} \\ &+c_{iy}\frac{F_i(\hat{\boldsymbol{x}} + \Delta y, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t})}{\Delta y} \\ &+c_{iz}\frac{F_i(\hat{\boldsymbol{x}} + \Delta z, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t})}{\Delta z} \\ = \quad &\frac{L}{U}(N_{\text{coll}})_i \end{aligned} \tag{2.21}$$

де $\Delta\hat{t} = \Delta t \cdot U/L$.

Підібравши значення кроку дискретизації по часу і простору таким чином, що виконувалась умова $\Delta\hat{\boldsymbol{x}}/\Delta\hat{t} = \boldsymbol{c}_i$, отримуємо таке рівняння:

$$\begin{aligned} &\frac{F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t})}{\Delta\hat{t}} + \frac{F_i(\hat{\boldsymbol{x}} + \boldsymbol{c}_i\Delta\hat{t}, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t} + \Delta\hat{t})}{\Delta\hat{t}} \\ = \quad &\frac{F_i(\hat{\boldsymbol{x}} + \boldsymbol{c}_i\Delta\hat{t}, \hat{t} + \Delta\hat{t}) - F_i(\hat{\boldsymbol{x}}, \hat{t})}{\Delta\hat{t}} = \frac{L}{U}(N_{\text{coll}})_i \end{aligned} \tag{2.22}$$

Прирівнявши $\Delta t = t_c$, та помноживши рівняння (2.22) на $\Delta \hat{t}$, отримуємо остаточну версію дискретного решітчастого рівняння Больцмана [192]:

$$F_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) - F_i(\boldsymbol{x}, t) = t_c (N_{\text{coll}})_i \tag{2.23}$$

Автори [192] так інтерпретують дане рівняння: «Це рівняння має конкретну просту фізичну інтерпретацію, в якій оператор зіткнення оцінюється локально, і існує лише один крок потоку або операція «зрушення» на швидкість решітки. Ця інтерпретація частинок потоку та зіткнення є наслідок лагранжевого характеру рівняння, для якого крок дискретизації решітки є відстанню, пройденою частинкою протягом кроку в часі. Дискретизації дискретного рівняння Больцмана вищого порядку зазвичай вимагають кількох операцій «зрушення» для оцінки кожної похідної, а інтерпретація частинок менш очевидна.»

2.1.5 Оператор зіткнень Бхатнагара - Гросса – Крука

Оператор Бхатнагара-Гросса-Крука (BGK) — це оператор зіткнення, який зазвичай використовується в решітчастій моделі Больцмана для моделювання потоку рідини. Він названий на честь своїх оригінальних розробників Прабху Л. Бхатнагара, Юджина П. Гросса та Макса Крука [92]. Оператор BGK вводить процес релаксації, який приводить функції розподілу до стану локальної рівноваги.

У кінетичному рівнянні Больцмана оператор зіткнення представляє взаємодію між частинками та враховує їх перерозподіл у просторі швидкостей. Оператор BGK спрощує термін зіткнення, припускаючи процес релаксації до локального розподілу рівноваги. Це спрощення дозволяє проводити ефективні чисельні обчислення та підтримує основну фізику зіткнень частинок.

Оператор BGK для решітчастого рівняння Больцмана виглядає наступним чином:

$$t_c (N_{\text{coll}})_i = -\frac{1}{\tau}\left(F_i - F_i^{(eq)}\right) \tag{2.24}$$

де τ – параметр релаксації, зв'язаний з в'язкістю рідини, контролює швидкість, з якою функції розподілу наближаються до стану локальної рівноваги, $F_i^{(eq)}$ - апроксимація локальної рівноважної функції розподілу.

Тоді формула (2.23) перетворюється в наступну:

$$F_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) - F_i(\boldsymbol{x}, t) = -\frac{1}{\tau}\left(F_i - F_i^{(eq)}\right) \tag{2.25}$$

2.1.6 Апроксимація рівноважної функції розподілу

В методі LBM, у формулі (2.24), рівноважна функція розподілу $F_i^{(eq)}$ виражається через розподіл Максвела-Больцмана [120]:

$$F^{eq}(\boldsymbol{v}, t) = \rho\left(\frac{m}{2\pi kT}\right)^{\frac{D}{2}} e^{-\frac{m(\boldsymbol{v}-\boldsymbol{u})^2}{2kT}} \tag{2.26}$$

де ρ - густина, D - розмірність обчислювального простору, k - стала Больцмана, T - температура, $\boldsymbol{v}$ – швидкість частинок рідини, $\boldsymbol{u}$ - швидкість рідини. Щоб спростити рівняння (2.26), використаємо зв'язок сталої Больцмана, температури та маси частинки зі швидкістю звуку в середовищі:

$$c_s^2 = \frac{kT}{m} \tag{2.27}$$

Щоб спростити рівняння (2.25) також використаємо ряд Тейлора для функції e^x, та формулу (2.27), маємо наступне рівняння:

$$\begin{aligned} e^{-\frac{(\boldsymbol{v}-\boldsymbol{u})^2}{2c_s^2}} &= e^{-\frac{v^2}{2c_s^2}} e^{-\frac{\boldsymbol{u}^2-2\boldsymbol{u}\cdot\boldsymbol{v}}{2c_s^2}} \approx e^{-\frac{\boldsymbol{v}^2}{2c_s^2}}\left(1 - \frac{\boldsymbol{u}^2 - 2\boldsymbol{u}\cdot\boldsymbol{v}}{2c_s^2} + \frac{(\boldsymbol{u}^2 - 2\boldsymbol{u}\cdot\boldsymbol{v})^2}{8c_s^4} + \cdots\right) \approx \\ &\approx e^{-\frac{\boldsymbol{v}^2}{2c_s^2}}\left(1 - \frac{\boldsymbol{u}^2}{2c_s^2} + \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{c_s^2} + \frac{(\boldsymbol{u}\cdot\boldsymbol{v})^2}{2c_s^4}\right) + o(\boldsymbol{u}^2) \end{aligned} \tag{2.28}$$

Остаточний вигляд рівноважна функція, на основі розподілу Максвела-Больцмана, для методу LBM має наступний:

$$F^{eq}(\boldsymbol{v}, t) = \frac{\rho}{(2\pi c_s^2)^{\frac{D}{2}}} e^{-\frac{\boldsymbol{v}^2}{2c_s^2}}\left(1 - \frac{\boldsymbol{u}^2}{2c_s^2} + \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{c_s^2} + \frac{(\boldsymbol{u}\cdot\boldsymbol{v})^2}{2c_s^4}\right) \tag{2.29}$$

2.1.7 Класифікація решітчастих моделей DnQm

У решітчастій моделі Больцмана вибір набору дискретних швидкостей частинок, або більш простіше решітки, відіграє вирішальну роль у дискретизації області рідини та моделюванні потоку рідини. Одним із часто використовуваних класів решіток є решітка *DnQm*, де *n* означає кількість вимірів (2 для двовимірного простору, 3 для тривимірного), а *m* позначає кількість дискретних швидкостей.

Решітки *DnQm* — це сітки, які визначають дискретні напрямки, в яких частинки поширюються та взаємодіють. Дискретні швидкості в кожному напрямку решітки ретельно вибираються для забезпечення певних властивостей, таких як ізотропія та інваріантність Галілея, а також для ефективної апроксимації безперервного простору швидкостей.

Дискретні швидкості в решітках *DnQm* представлені векторами $\boldsymbol{c_i} = (c_{i1}, c_{i2}, \ldots, c_{in})$, де кожен компонент c_{ij} відповідає швидкості вздовж *j*-го виміру. Решітка *DnQm* визначає набір цих векторів швидкостей для заданої кількості вимірів і дискретних швидкостей.

Приведемо кілька прикладів решіток, які найчастіше використовуються для моделювання руху рідин у дво- та тривимірному просторі, та наведемо формули для обчислення всіх дискретних швидкостей в кожній решітці. Візуалізації решіток зображені на рис. 2.2 та 2.3

- **D2Q4**

$$\boldsymbol{c}_i = \left(\cos\frac{2\pi i}{4}, \sin\frac{2\pi i}{4}\right)\ i = 1,2,3,4 \tag{2.30}$$

- **D2Q9**

$$\begin{aligned} \boldsymbol{c}_0 &= (0,0), \\ \boldsymbol{c}_{1,3}, \boldsymbol{c}_{2,4} &= (\pm 1,0),\ (0,\pm 1) \\ \boldsymbol{c}_{5,6,7,8} &= (\pm 1,\pm 1) \end{aligned} \tag{2.31}$$

- **D3Q15**

$$\begin{aligned} \boldsymbol{c}_0 &= (0,0,0) \\ \boldsymbol{c}_{1,2}, \boldsymbol{c}_{3,4}, \boldsymbol{c}_{5,6} &= (\pm 2,0,0), (0,\pm 2,0)(0,0,\pm 2) \\ \boldsymbol{c}_{7,\ldots,14} &= (\pm 1,\pm 1,\pm 1) \end{aligned} \tag{2.32}$$

- **D2Q13**

$$\begin{aligned} \boldsymbol{c}_0 &= (0,0) \\ \boldsymbol{c}_{1,2}, \boldsymbol{c}_{3,4} &= (\pm 1,0), (0,\pm 1) \\ \boldsymbol{c}_{5,6,7,8} &= (\pm 1,\pm 1) \\ \boldsymbol{c}_{9,10}, \boldsymbol{c}_{11,12} &= (\pm 2,0), (0,\pm 2) \end{aligned} \tag{2.33}$$

- **D3Q19**

$$\begin{aligned} \boldsymbol{c}_0 &= (0,0) \\ \boldsymbol{c}_{1,2}, \boldsymbol{c}_{3,4}, \boldsymbol{c}_{5,6} &= (\pm 1,0,0), (0,\pm 1,0)(0,0,\pm 1) \\ \boldsymbol{c}_{7,\ldots,10}, \boldsymbol{c}_{11,\ldots,14}, \boldsymbol{c}_{15,\ldots,18} &= (\pm 1,\pm 1,0), (\pm 1,0,\pm 1), (0,\pm 1,\pm 1) \end{aligned} \tag{2.34}$$

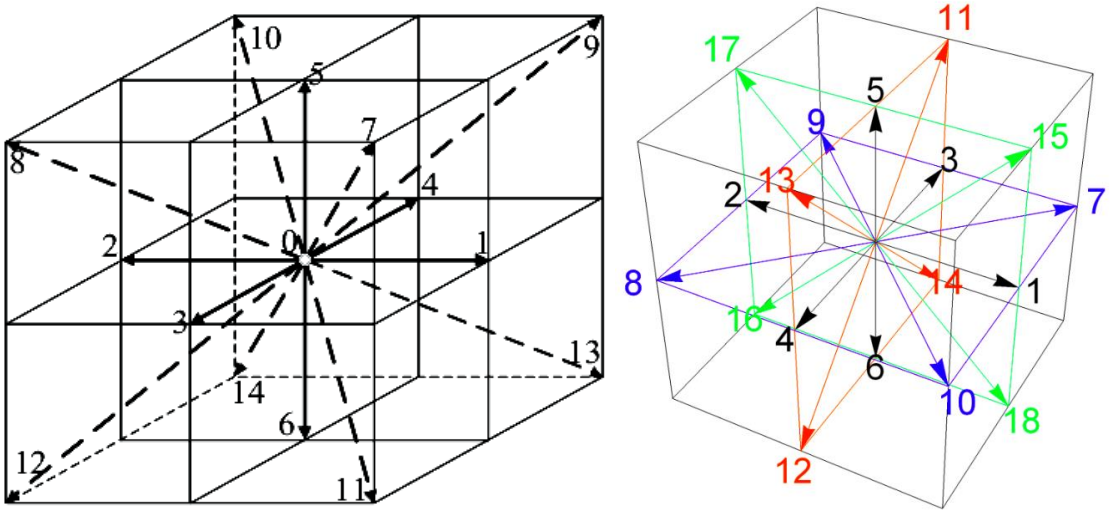

Рисунок 2.2. Зліва – решітка *D3Q15*, справа – *D3Q19*.

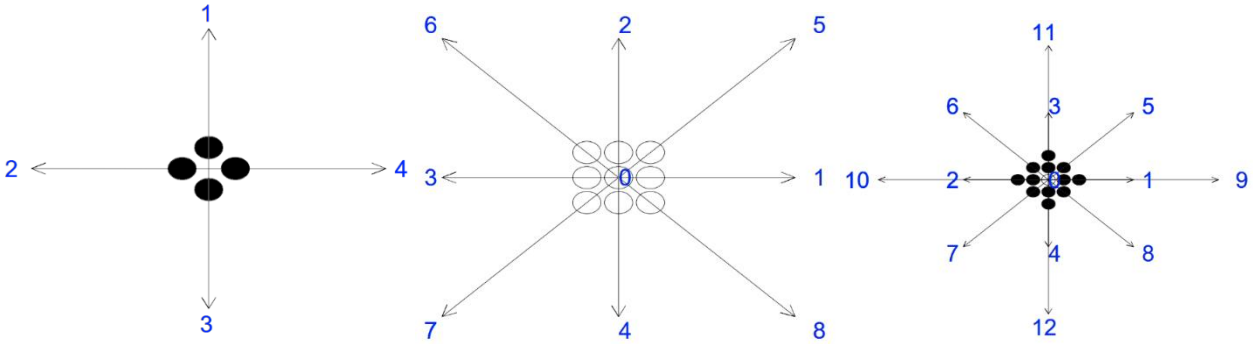

Рисунок 2.3. Зверху зліва – решітка *D2Q4*, зверху справа - *D2Q9*, знизу - *D2Q13*

В цій роботі експерименти проведені у двовимірному просторі, із застосуванням решітки *D2Q9*, тому в подальшому буде розглядатись тільки ця решітка, оскільки вона проста в реалізації, вимагає менший об’єм обчислень, в порівнянні зі складнішими схемам, та має прийнятну точність.

2.1.8 Решітчаста модель Больцмана у двовимірному просторі

Розглянемо двовимірну обчислювальну область. Розіб'ємо її на рівні за розміром комірки, таким чином, щоб кількість комірок була $N \times N$, розміри комірки $\Delta x \times \Delta y$, при чому $\Delta x = \Delta y$. Введемо величину c – швидкість частинок у дискретизованому просторі:

$$c = \frac{\Delta x}{\Delta t} \tag{2.35}$$

де Δt – крок дискретизації по часу.

Швидкість звуку дорівнює [122]:

$$c_s^2 = \frac{c^2}{3} \tag{2.36}$$

Оператор BGK, за допомогою параметра релаксації τ, встановлює значення в'язкості рідини ν. Такий зв'язок встановлюється наступною формулою [193]:

$$\nu = c_s^2 \Delta t \left(\tau - \frac{1}{2}\right) \tag{2.37}$$

Решітка *D2Q9* містить 9 дискретних напрямків $\boldsymbol{c_i}$, відповідно кожна комірка обчислювальної області має 9 значень функції розподілу частинок $F_i, i = 0,1,..,8$. Кожна з цих функцій визначає ймовірність певної із 9 швидкостей $\boldsymbol{v_i}$, яку може мати частинка. При цьому $\boldsymbol{v_i} = \boldsymbol{c_i} \cdot c$.

Застосувавши до рівняння (2.28) принципу максимальної ентропії за обмежень збереження маси та імпульсу [194], отримаємо рівняння рівноважної функції в наступному вигляді:

$$F_i^{\mathrm{eq}} = w_i \rho \left(1 + \frac{\boldsymbol{v_i} \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{v_i} \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right) \tag{2.38}$$

де w_i – ваговий коефіцієнт. Для решітки *D2Q9* вагові коефіцієнти будуть наступні [194]:

$$w_i = \begin{cases} \frac{4}{9}, i = 0 \\ \frac{1}{9}, i = 1,2,3,4 \\ \frac{1}{36}, i = 5,6,7,8 \end{cases} \quad (2.39)$$

Класичний алгоритм для одного часового кроку в кожній комірці обчислювальної області методу LBM, для розглянутої вище моделі *D2Q9* має два кроки [112]:

1. **Крок зіткнення**

   Обчислення макроскопічних величин густини ρ (2.41) та швидкості $\boldsymbol{u}$ (2.42), обчислення рівноважної функції розподілу F_i^{eq}, за допомогою формули (2.38), та оновлення функції розподілу за допомогою оператора зіткнення BGK та формули (2.25)

2. **Крок поширення**

   На етапі поширення оновлені функції розподілу поширюються на сусідні вузли решітки відповідно до їх дискретних швидкостей. Цей крок передбачає рух частинок уздовж відповідних векторів швидкостей:

$$F_i(\boldsymbol{x} + \boldsymbol{v}_i, t + \Delta t) = F_i(\boldsymbol{x}, t + \Delta t) \quad (2.40)$$

Обчислення макроскопічних величин густини ρ та швидкості $\boldsymbol{u}$ відбувається за наступними формулами:

$$\rho(\boldsymbol{x}, t) = \sum_{i=0}^{8} F_i(\boldsymbol{x}, t) \quad (2.41)$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = \frac{1}{\rho(\boldsymbol{x}, t)} \sum_{i=0}^{8} \boldsymbol{v}_i F_i(\boldsymbol{x}, t) \quad (2.42)$$

2.1.9 Перехід від рівняння Больцмана до рівняння Нав'є-Стокса

Для підтвердження доцільності використання решітчастого рівняння Больцмана для моделювання руху рідин, покажемо зв'язок між цим рівнянням та

рівнянням Нав'є-Стокса. Вперше такий зв'язок був встановлений у роботі 1997 року [122], за допомогою розкладу Чепмена-Енського.

Розкладемо функцію розподілу, за допомогою розкладу Чепмена-Енського на наступні вирази:

$$F_i = F_i^{eq} + KF_i^{neq} \tag{2.43}$$

$$F_i^{neq} = F_i^{(1)} + KF_i^{(2)} + O(K^2) \tag{2.44}$$

де F_i^{neq} – нерівноважна складова, K – число Кнудсена.

Зв'язок між макроскопічними величинами та рівноважними та нерівноважними розподілами наступний:

$$\rho = \sum_i F_i^{\mathrm{eq}} \tag{2.45}$$

$$\rho \boldsymbol{u} = \sum_i F_i^{\mathrm{eq}} \boldsymbol{v}_i \tag{2.46}$$

$$0 = \sum_i F_i^{(k)}, k = 1{,}2 \tag{2.47}$$

$$0 = \sum_i F_i^{(k)} \boldsymbol{v_i} \tag{2.48}$$

З урахуванням формул вище, розклад Чепмана-Енського буде наступний:

$$\frac{\partial}{\partial t} = K\frac{\partial}{\partial t_1} + K^2\frac{\partial}{\partial t_2}, t_2 \ll t_1 \tag{2.49}$$

$$\frac{\partial}{\partial x} = K\frac{\partial}{\partial x_1} \tag{2.50}$$

Представимо рівняння (2.25) у вигляді розкладу Тейлора:

$$\frac{\partial F_i}{\partial t} + \boldsymbol{v}_i \cdot \nabla F_i + \left(\frac{1}{2}\boldsymbol{v}_i\boldsymbol{v}_i : \nabla\nabla F_i + \boldsymbol{v}_i \cdot \nabla\frac{\partial F_i}{\partial t} + \frac{1}{2}\frac{\partial^2 F_i}{\partial t^2}\right) = \frac{1}{\tau}\left(F_i^{eq} - F_i\right) \tag{2.51}$$

Отримаємо рівняння неперервності шляхом підстановки розширених рівноважних і нерівноважних функцій розподілу у розклад Тейлора (2.51), а потім виділимо різні частини в залежності від порядку K.

K^0 :

$$\frac{\partial F_i^{\mathrm{eq}}}{\partial t_1} + \boldsymbol{v}_i\nabla_1 F_i^{\mathrm{eq}} = -\frac{F_i^{(1)}}{\tau} \tag{2.52}$$

K^1 :

$$\frac{\partial F_i^{(1)}}{\partial t_1} + \frac{\partial F_i^{\mathrm{eq}}}{\partial t_2} + \boldsymbol{v}_i \nabla F_i^{(1)} + \frac{1}{2}\boldsymbol{v}_i \boldsymbol{v}_i : \nabla\nabla F_i^{\mathrm{eq}} + \boldsymbol{v}_i \cdot \nabla \frac{\partial F_i^{\mathrm{eq}}}{\partial t_1} + \frac{1}{2}\frac{\partial^2 F_i^{\mathrm{eq}}}{\partial t_1^2} = -\frac{F_i^{(2)}}{\tau} \quad (2.53)$$

Спростимо рівняння (2.53):

$$\frac{\partial F_i^{\mathrm{eq}}}{\partial t_2} + \left(1 - \frac{1}{2\tau}\right)\left[\frac{\partial F_i^{(1)}}{\partial t_1} + \boldsymbol{v}_i \nabla_1 f_i^{(1)}\right] = -\frac{F_i^{(2)}}{\tau} \quad (2.54)$$

Застосувавши рівняння (2.45-2.48) отримаємо:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{u} = 0 \quad (2.55)$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot \Pi = 0 \quad (2.56)$$

Таким чином ми отримали рівняння неперервності для маси та імпульсу.

Тензор потоку імпульсу Π:

$$\Pi_{xy} = \sum_i \boldsymbol{v}_{ix}\boldsymbol{v}_{iy}\left[F_i^{eq} + \left(1 - \frac{1}{2\tau}\right)F_i^{(1)}\right] \quad (2.57)$$

де $\boldsymbol{v}_{ix}\boldsymbol{v}_{iy} = \left(\sum_x \boldsymbol{v}_{ix}\right)^2$

Підставивши значення рівноважної функції розподілу у рівняння (2.57) отримаємо:

$$\Pi_{xy}^{(0)} = \sum_i \boldsymbol{v}_{ix}\boldsymbol{v}_{iy}F_i^{eq} = p\Delta\boldsymbol{x} + \rho u_x u_y \quad (2.58)$$

$$\Pi_{xy}^{(1)} = \left(1 - \frac{1}{2\tau}\right)\sum_i \boldsymbol{v}_{ix}\boldsymbol{v}_{iy}F_i^{(1)} = \nu\left(\nabla_x\left(\rho \boldsymbol{u}_y\right) + \nabla_y(\rho \boldsymbol{u}_x)\right) \quad (2.59)$$

Припустимо, що коливання значень густини рідини незначні, отримуємо рівняння Нав'є-Стокса [129]:

$$\rho\left(\frac{\partial \vec{u}_x}{\partial t} + \nabla_y \cdot \vec{u}_x \vec{u}_y\right) = -\nabla_x p + \nu\nabla_y \cdot \left(\nabla_x\left(\rho\vec{u}_y\right) + \nabla_y(\rho\vec{u}_x)\right) \quad (2.60)$$

Таким чином ми встановили зв'язок між решітчастою моделлю Больцмана та рівнянням Нав'є-Стокса.

2.1.10 Початкові умови

У методах LBM початкові умови відносяться до специфікації властивостей рідини на початку моделювання. Ці початкові умови необхідні для ініціалізації функцій розподілу LBM і визначення початкового стану рідини.

У LBM рідина представлена на дискретній решітці, а функції розподілу визначені в кожній точці ґратки. Для задання початкових умов функціям розподілу присвоюють певні значення, які відповідають бажаному початковому стану рідини.

Вибір початкових умов залежить від конкретної задачі, що моделюється. Деякі початкові умови, які зазвичай використовуються в LBM, включають:

- Ініціалізація функцій розподілу рівноважними розподілами. Ці розподіли визначаються бажаними макроскопічними властивостями рідини, такими як густина та швидкість. Встановлюючи функції розподілу на рівноважні значення, рідина починає перебувати в стані спокою або рівномірного потоку.
- В деяких випадках початкові умови можуть бути визначені за допомогою аналітичних виразів. Наприклад, для потоків із специфічними профілями, такими як параболічний профіль швидкості в потоці труби, функції розподілу можна ініціалізувати на основі цих відомих профілів.
- В окремих ситуаціях початкові умови також можна отримати з експериментальних вимірювань. У цьому випадку функції розподілу встановлюються на основі виміряних значень властивостей рідини, таких як швидкість або щільність, у різних місцях області.
- Для моделювання специфічних проблем може бути бажаним ввести деяку випадковість або збурення у початкові умови, щоб імітувати турбулентні або нестаціонарні потоки. Випадкові числа або специфічні шаблони збурень можуть бути використані для встановлення функцій розподілу, що призводить до нерівномірного початкового стану.

Важливо зазначити, що вибір початкових умов може значно вплинути на поведінку та точність моделювання. Вибрані початкові умови повинні

представляти фізично реалістичну початкову точку для рідини, гарантуючи, що подальша еволюція системи відповідає бажаній поведінці.

Розглянемо метод задання початкових умов, при якому на всій обчислювальній області значення макроскопічної швидкості $\boldsymbol{u}$ задається певною функцією $\boldsymbol{u}_{init}(\boldsymbol{x})$, яка для кожної точко простору $\boldsymbol{x}$, ставить у відповідність певне значення швидкості. Аналогічним чином, за допомогою функції $\rho_{init}(\boldsymbol{x})$ задамо значення густини ρ.

Використавши формулу рівноважної функції розподілу (2.38), дискретних векторів решітки *D2Q9* (2.31) та вагових коефіцієнтів (2.39), виведемо формули для функцій розподілу на основі початкових значень $\boldsymbol{u}$:

$$
\begin{aligned}
F_0 &= \frac{4}{9}\left(1 - \frac{3}{2c^2}u_x^2 - \frac{3}{2c^2}u_y^2\right), \\
F_1 &= \frac{1}{9}\left(1 + \frac{3}{c}u_x + \frac{3}{c^2}u_x^2 - \frac{3}{2c^2}u_y^2\right) \\
F_2 &= \frac{1}{9}\left(1 + \frac{3}{c}u_y + \frac{3}{c^2}u_y^2 - \frac{3}{2c^2}u_x^2\right) \\
F_3 &= \frac{1}{9}\left(1 - \frac{3}{c}u_x + \frac{3}{c^2}u_x^2 - \frac{3}{2c^2}u_y^2\right) \\
F_4 &= \frac{1}{9}\left(1 - \frac{3}{c}u_y + \frac{3}{c^2}u_y^2 - \frac{3}{2c^2}u_x^2\right) \\
F_5 &= \frac{1}{36}\left(1 + \frac{3}{c}u_x + \frac{3}{c}u_y + \frac{3}{c^2}u_x^2 + \frac{3}{c^2}u_y^2 + \frac{9}{c^2}u_x u_y\right) \\
F_6 &= \frac{1}{36}\left(1 - \frac{3}{c}u_x + \frac{3}{c}u_y + \frac{3}{c^2}u_x^2 + \frac{3}{c^2}u_y^2 - \frac{9}{c^2}u_x u_y\right) \\
F_7 &= \frac{1}{36}\left(1 - \frac{3}{c}u_x - \frac{3}{c}u_y + \frac{3}{c^2}u_x^2 + \frac{3}{c^2}u_y^2 + \frac{9}{c^2}u_x u_y\right) \\
F_8 &= \frac{1}{36}\left(1 + \frac{3}{c}u_x - \frac{3}{c}u_y + \frac{3}{c^2}u_x^2 + \frac{3}{c^2}u_y^2 - \frac{9}{c^2}u_x u_y\right)
\end{aligned}
\tag{2.61}
$$

2.1.11 Граничні умови

Граничні умови в методах LBM стосуються специфікації того, як рідина взаємодіє з межами обчислювальної області під час моделювання. Ці умови

відіграють вирішальну роль у фіксуванні поведінки рідини поблизу кордонів і забезпеченні точності та стабільності моделювання LBM.

LBM використовує схему відскоку для обробки граничних умов, де функції розподілу модифікуються на границях для врахування взаємодії з твердими поверхнями. Існує кілька поширених типів граничних умов, що використовуються в LBM [121, 195, 196]:

- Гранична умова відсутності ковзання: у цій умові швидкість рідини на межі дорівнює нулю, імітуючи ефект суцільної стінки. Функції розподілу, що входять в область від кордону (вхідні розподіли), відбиваються назад до протилежних швидкостей решітки (вихідні розподіли), ніби відскакуючи від стіни. Ця умова представляє типовий сценарій, коли частинки рідини прилипають до граничної поверхні та не ковзають повз неї.
- Гранична умова вільного ковзання: ця умова передбачає, що швидкість рідини на межі паралельна поверхні, але між рідиною та межею немає напруги зсуву. Вхідні розподіли відображаються назад до вихідних розподілів, але зі зворотними компонентами швидкості, нормальними до границі. Ця умова часто використовується для імітації потоку повз непроникні межі з мінімальним тертям.
- Граничні умови притоку/витоку: у моделюванні LBM зазвичай мають задані умови притоку чи витоку на певних границях. Для припливу функції розподілу на межі встановлюються на задані значення на основі бажаних умов входу. Для відтоку розподіли зазвичай визначають шляхом екстраполяції з внутрішньої частини області. Ці умови гарантують, що рідина входить або виходить із обчислювальної області з бажаними властивостями.

Далі будуть розглянуті та описані типи граничних умов, які були використані в розробленому методі при проведені експериментів, що описані в розділі 5.

Гранична умова притоку

Розглянемо випадок, зображений на рис. 2.4, на лівій границі швидкість вхідного потоку дорівнює $\boldsymbol{u} = (u_x, 0), u_x = U_x$.В цій ситуації значення густини ρ та функцій розподілу F_1, F_5 та F_8 є невідомими, і їх необхідно розрахувати для всіх

комірок на лівій границі обчислювальної області. Використавши формулу (2.41) для макроскопічної густини, введемо наступні змінні для комірки на лівій границі:

$$\rho_1 = F_1 + F_5 + F_8 \tag{2.62}$$

$$\rho_2 = F_0 + F_2 + F_4 \tag{2.63}$$

$$\rho_3 = F_3 + F_6 + F_7 \tag{2.64}$$

$$\rho = \rho_1 + \rho_2 + \rho_3 \tag{2.65}$$

Значення ρ_2 та ρ_3 – відомі, тоді як ρ_1 є невідомим. Використавши формулу (2.42) та той факт, що y-складова швидкості притоку $\boldsymbol{u}$ дорівнює нулю отримаємо наступний вираз:

$$\rho u_x = \rho_1 - \rho_3 \tag{2.66}$$

Значення густини ρ можна виразити через відомі значення швидкості вхідного потоку u_x, ρ_2 та ρ_3, використавши формулу (2.65):

$$\rho = \frac{\rho_2 + 2\rho_3}{1 - u_x} \tag{2.67}$$

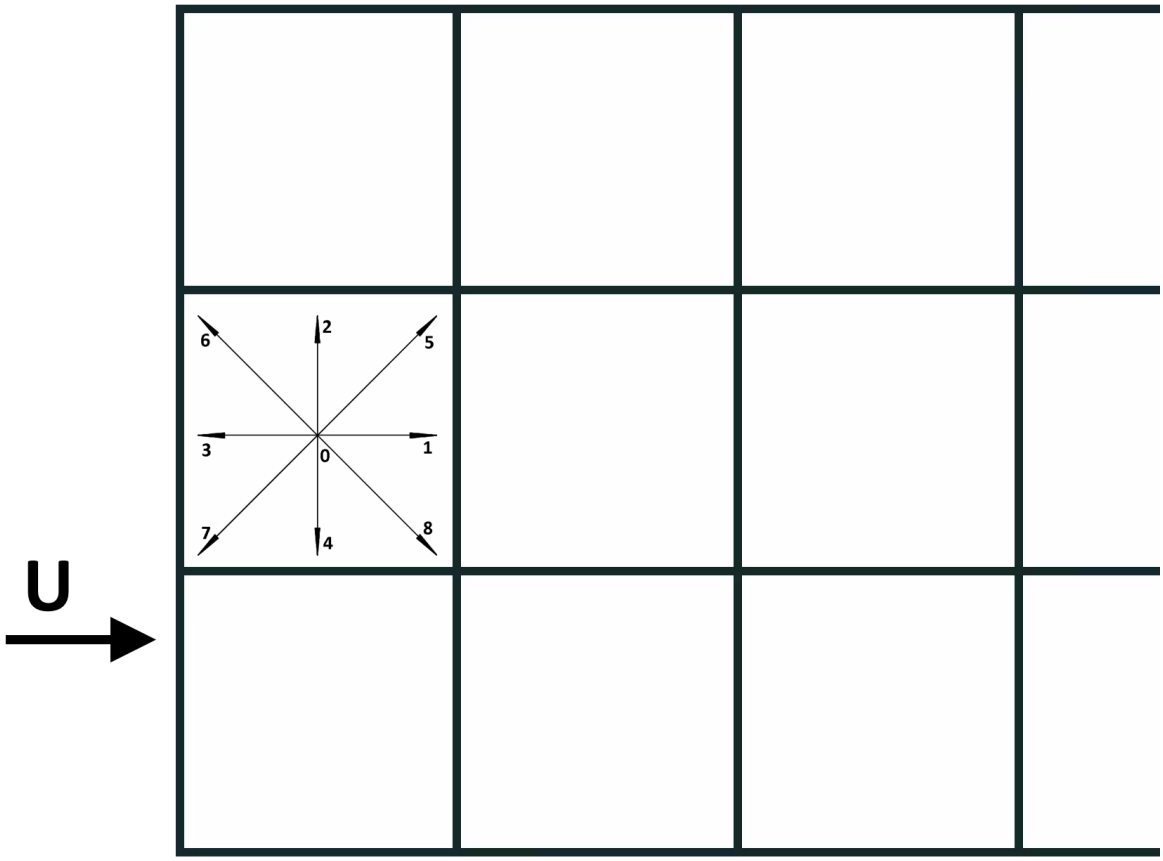

Рисунок 2.4 Гранична умова притоку рідини

Для знаходження невідомих значень функцій розподілу F_1, F_5 та F_8, використаємо співвідношення розроблене в роботі [121]:

$$F_i - F_i^{eq} = F_{\bar{i}} - F_{\bar{i}}^{eq} \tag{2.68}$$

де $\bar{i}$ – напрям швидкості, протилежний i. Тоді невідомі функції розподілу можна вирахувати наступними формулами:

$$F_1 = F_1^{eq} + F_3 - F_3^{eq} \tag{2.69}$$

$$F_5 = F_5^{eq} + F_7 - F_7^{eq} \tag{2.70}$$

$$F_8 = F_8^{eq} + F_6 - F_6^{eq} \tag{2.71}$$

Описані вище формули також можна застосувати у випадках, коли вхідний потік прибуває не тільки з лівої сторони обчислювальної області, але й з інших сторін. Єдиною відмінністю буде те, що функції розподілу значення яких на границях невідомі, будуть мати інші індекси.

Гранична умова витоку

Для граничної умови витоку ми використали метод, який називається проста гранична умова екстраполяції [197]. Його суть полягає в копіюванні значень функцій розподілу, що входять направлені ззовні обчислювальної області, з останньої комірки, що знаходиться перед коміркою на границі витоку рідини. Для детальнішого пояснення розглянемо граничну умову на рис. 2.5

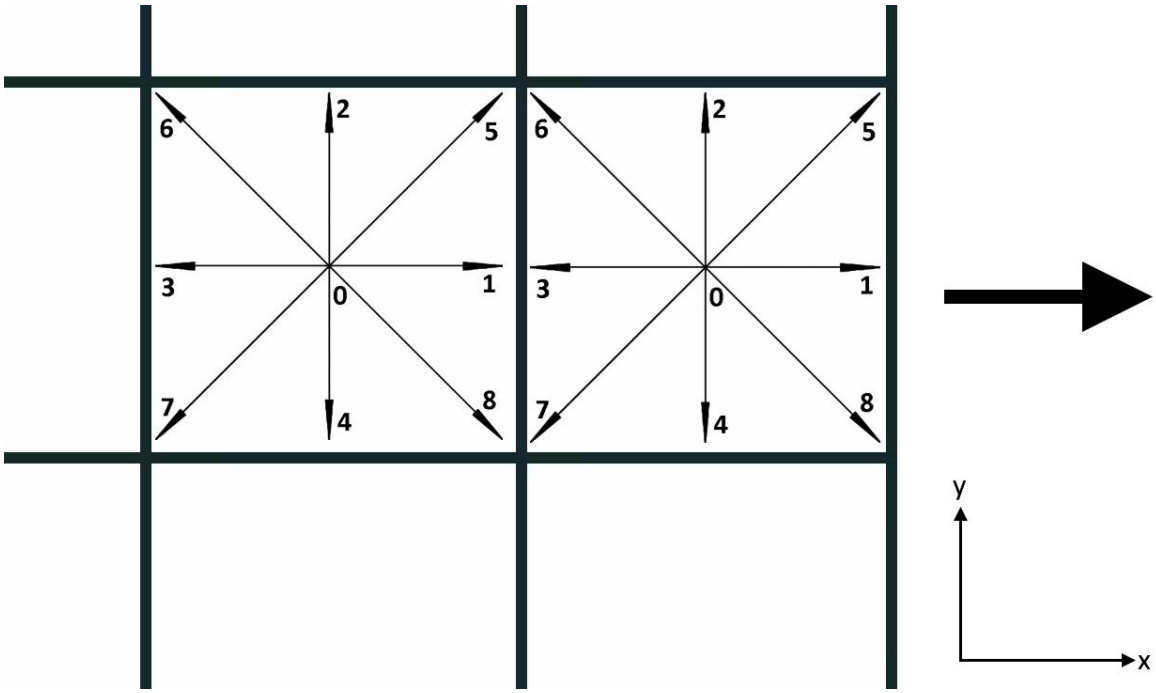

Рисунок 2.5 Гранична умова витоку рідини

В цій ситуації витік рідини відбувається через праву сторону обчислювальної області. Тому значення функцій розподілу F_3, F_6 та F_7 є невідомими. Нехай $F_i(x, y)$ – значення функції розподілу для напрямку i, в точці простору (x, y). Тоді при застосуванні простої граничної умови екстраполяції, невідомі значення функцій розподілу в граничній комірці будуть наступні:

$$F_3(x, y) = F_3(x - \Delta x, y) \tag{2.72}$$

$$F_6(x, y) = F_6(x - \Delta x, y) \tag{2.73}$$

$$F_7(x, y) = F_7(x - \Delta x, y) \tag{2.74}$$

де Δx – крок дискретизації області по осі X.

Як і у випадку з умовою притоку, дані формули можна застосувати і для випадків, коли витік відбувається з іншої сторони обчислювальної області, визначивши функції розподілу, напрямки яких приходять в обчислювальну область.

Гранична умова твердої стіни

Для моделювання складних геометрій та обчислювальних областей, при моделюванні методами LBM, комірки діляться на 2 типи – ті, що містять рідину, і ті, куди рідина не потрапляє, так звана «стінка» або «суха комірка».

Під час моделювання в'язкої рідини на нерухомих твердих границях, внаслідок сил молекулярної взаємодії між рідиною та твердим тілом використовується умова прилипання рідини. Умова прилипання рідини визначається таким чином:

$$\boldsymbol{u} = 0 \tag{2.75}$$

Для реалізації цієї граничної умови ми використали згадану в розділі 1.3 умову зворотного відображення. Коли функція розподілу досягає граничного вузла, вона відбивається назад у протилежному напрямку швидкості решітки. Це відображення здійснюється шляхом заміни функцій розподілу, що входять у граничний вузол (вхідні розподіли), на розподіли, що виходять із граничного вузла (вихідні розподіли), але зі зворотними швидкостями, нормальними до границі рідкого і твердого середовищ (рис 2.6). Умова зворотного відображення є простим і обчислювально ефективним методом обробки взаємодії рідини та твердого тіла в

методі LBM. Вона може бути легко реалізована і застосовна до широкого діапазону граничних геометрій.

Умову зворотного відображення має наступне математичне формулювання:

$$F_{\bar{i}}(\boldsymbol{x}, t) = F_i(\boldsymbol{x}, t), \boldsymbol{x} \in \Omega \tag{2.76}$$

де Ω – область, що містить тверде тіло.

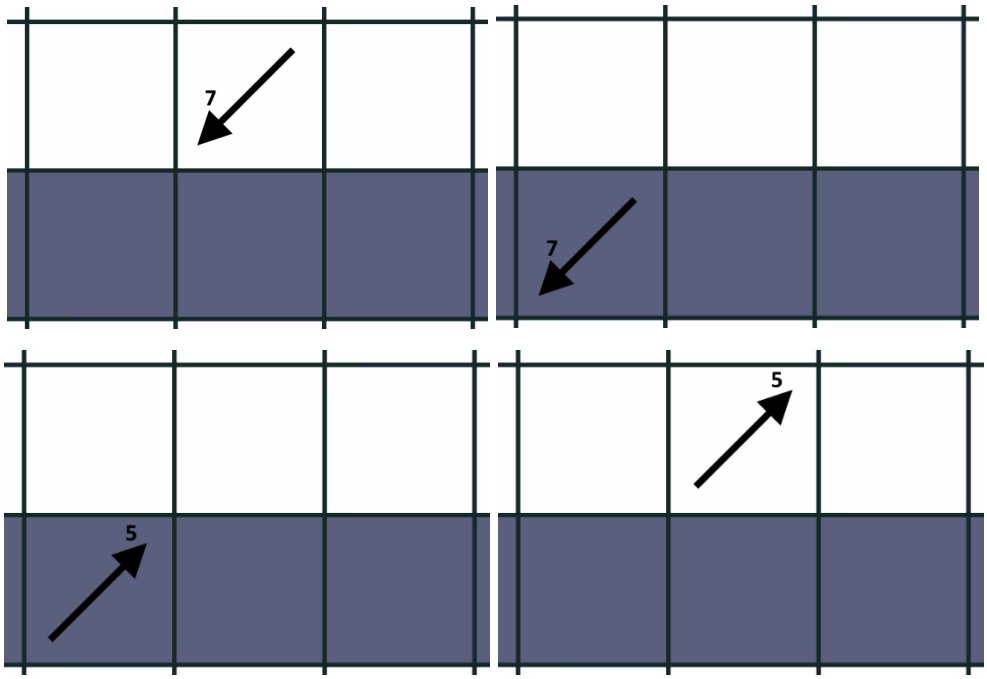

Рисунок 2.6 – Ілюстрація умови зворотного відображення. Зверху – момент часу t, знизу – момент часу , $t + \Delta t$. Зліва - крок зіткнення, справа – крок поширення

Для схеми решітки *D2Q9* зворотнє відображення функцій розподілу буде мати вигляд:

$$F_1 = F_3; F_2 = F_4; F_5 = F_7; F_6 = F_8 \tag{2.76}$$

Виведемо формули для компонент макроскопічної швидкості $\boldsymbol{u}_\Omega$, для комірок твердого тіла, використовуючи формулу (2.42):

$$u_{\Omega x} = \frac{(f_1 + f_5 + f_8 - f_3 - f_6 - f_7)}{\rho} \tag{2.77}$$

$$u_{\Omega y} = \frac{c(f_4 + f_7 + f_8 - f_2 - f_5 - f_6)}{\rho} \tag{2.78}$$

Враховуючи значення (2.76), маємо $u_{\Omega x} = 0$, $u_{\Omega y} = 0$. Отже, умова зворотного відображення є достатньою для реалізації прилипання рідини до твердого тіла під час моделювання методом LBM.

2.1.12 Модифікована рівноважна функція розподілу

Незважаючи на всі переваги методу LBM, деякі автори [199,200] вказують на чисельну нестабільність традиційних методів LBM, і пропонують альтернативні підходи до побудови кінетичних моделей, на основі яких працюють методи LBM. Тому ми використали модифіковану рівноважну функцію розподілу, на основі мінімізації дискретної ентропії, для забезпечення безумовної лінійної стабільності [200].

Нехай стан рівноваги це мінімізатор функціоналу Ляпунова H [201]:

$$H = \sum_{i=0}^{8} h_i(F_i) \tag{2.79}$$

де h_i – випукла функція, що обмежена формулами (2.41) і (2.42).

Враховуючи формули (2.41) і (2.42), введемо наступне рівнння:

$$\sum_{i=0}^{8} (h_i(F_i) - \lambda_0 F_i - \lambda_\alpha v_{i\alpha} F_i) = 0 \tag{2.80}$$

де $\lambda_0, \lambda_\alpha$ – множники Лагранжа (?). Похідна функції h_i по F_i^{eq} буде:

$$h_i'\left(F_i^{\mathrm{eq}}\right) = \lambda_0 + \lambda_\alpha v_{i\alpha} \tag{2.81}$$

Позначимо $\mu_i = \left[h_i'\left(F_i^{\mathrm{eq}}\right)\right]^{-1}$, тоді:

$$F_i^{eq} = \mu_i(\lambda_0 + \lambda_\alpha v_{i\alpha}) \tag{2.82}$$

Враховуючи попередні формули, функція H буде дорівнювати:

$$H = \sum_{i=0}^{8} F_i \ln\left(\frac{F_i}{w_i}\right) \tag{2.83}$$

Відповідно, рівноважна функція розподілу буде дорівнювати:

$$F_i^{\mathrm{eq}} = w_i \exp(\lambda_0) \prod_{\alpha=1}^{2} \exp(v_{i\alpha}\lambda_\alpha) \tag{2.84}$$

Позначимо $X = \exp(-\lambda_0)$ та $Z_\alpha = \exp(\lambda_\alpha)$, та запишемо рівняння рівноважної функції розподілу наступним чином:

$$F_i^{\mathrm{eq}} = w_i X^{-1} \prod_{\alpha=1}^{2} Z_\alpha^{v_{i\alpha}} \tag{2.85}$$

Тоді, рівняння для макроскопічних величин набувають вигляду:

$$\rho X = \sum_{i=0}^{8} w_i \prod_{\alpha=1}^{2} Z_\alpha^{v_{i\alpha}} \tag{2.86}$$

$$\rho u_\beta X = \sum_{i=0}^{8} w_i v_{i\beta} \prod_{\alpha=1}^{2} Z_\alpha^{v_{i\alpha}}, \beta = 1{,}2 \tag{2.87}$$

Перетворивши рівняння (2.86) та (2.87) у систему лінійних рівнянь, та розв’язавши її, отримаємо:

$$Z_\alpha = \frac{2u_\alpha + \sqrt{\left(\frac{u_\alpha}{c_s}\right)^2 + 1}}{1 - u_\alpha} \tag{2.88}$$

$$X^{-1} = \rho \prod_{\alpha=1}^{2} \left(2 - \sqrt{\left(\frac{u_\alpha}{c_s}\right)^2 + 1}\right) \tag{2.89}$$

В результаті, ми отримали модифіковану рівноважну функцію розподілу, яку використали в подальшій роботі:

$$F_i^{\mathrm{eq}} = w_i \rho \prod_{\alpha=1}^{2} \left(2 - \sqrt{\left(\frac{u_\alpha}{c_s}\right)^2 + 1}\right) \left(\frac{2u_\alpha + \sqrt{\left(\frac{u_\alpha}{c_s}\right)^2 + 1}}{1 - u_\alpha}\right)^{v_{i\alpha}} \tag{2.90}$$

2.1.13 Уточнення поля швидкості за допомогою рівняння Пуассона для тиску

Алгоритми, що ґрунтуються на LBM, припускають, що відбувається моделювання стислої рідини. У такому випадку тиск є функцією густини в методах LBM, незалежно від визначеного рівняння стану. Коли ми моделюємо стискаючі потоки, цей аспект не викликає проблем. Але коли ми працюємо з моделюванням нестисливих рідин, потрібно скорегувати похибку, яка виникає від коливання густини. Щоб визначити динаміку зміни густини та швидкості рідини, ми

скористаємося висновками [129], що підтверджують, що розв'язок крайової задачі на основі рівняння Нав'є-Стокса збігається з розв'язком рівняння Больцмана при малих значеннях числа Маха:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \\ \dfrac{\partial \rho u}{\partial t} + \nabla \cdot (u \otimes u) = -\dfrac{\nabla p}{\rho} + \dfrac{1}{\rho} \nabla \cdot (\bar{\tau}) + f \end{cases} \tag{2.91}$$

Визначимо тензор в’язких напружень $\bar{\tau}$ для стисливої ньютонівської рідини через поле швидкостей рідин:

$$\bar{\tau} = 2\mu \left[\frac{\nabla u + (\nabla u)^T}{2} - \frac{1}{3} (\nabla u) \bar{I} \right] \tag{2.92}$$

При малому значенні числа Маха при переході до нестисливої крайової задачі рівняння (2.92) можна виразити наступним чином [225]:

$$\frac{1}{\rho} \nabla \cdot (\bar{\tau}) = \upsilon \Delta u \tag{2.93}$$

Для визначення розподілу тиску на кожній часовій ітерації визначаємо рівняння Пуассона для тиску [225]:

$$\nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p \right) = \nabla u^* \tag{2.94}$$

У зв’язку з тим, що ми нехтуємо стисливістю досліджуваних рідин, ми можемо застосувати лінійну дискретизацію правої частини рівняння (2.94) за допомогою центральної різниці. Тоді

$$\frac{\partial}{\partial t} \left(\frac{\partial \tilde{u}_{i,j}}{\partial x} \right) = \frac{1}{\Delta t} \left(\frac{\tilde{u}_{i+1,j} - \tilde{u}_{i-1,j}}{2\Delta x} + \frac{\tilde{u}_{i,j+1} - \tilde{u}_{i,j-1}}{2\Delta y} \right) \tag{2.95}$$

де $\Delta x, \Delta y, \Delta t$ — кроки по просторових та часових координатах.

У відповідності з викладками, наведеними в [202], дискретизуємо ліву частину (2.94) за допомогою схеми скінченного об’єму:

$$p_{i,j}^n = \frac{1}{G_{i,j}} \left(\frac{p_{i+1,j}^n}{\rho_{\frac{i+1}{2},j}} + \frac{p_{i-1,j}^n}{\rho_{\frac{i-1}{2},j}} + \frac{p_{i,j+1}^n}{\rho_{i,\frac{j+1}{2}}} + \frac{p_{i,j-1}^n}{\rho_{i,\frac{j-1}{2}}} - S_{i,j} \right) \tag{2.96}$$

де $G_{i,j} = \frac{1}{\rho_{\frac{i+1}{2},j}} + \frac{1}{\rho_{\frac{i-1}{2},j}} + \frac{1}{\rho_{i,\frac{j+1}{2}}} + \frac{1}{\rho_{i,\frac{j-1}{2}}}$, $\rho_{\frac{i\pm1}{2},j} = \frac{\rho_{i,j}+\rho_{i\pm1,j}}{2}$, $\rho_{i,\frac{j\pm1}{2}} = \frac{\rho_{i,j}+\rho_{i,j\pm1}}{2}$, $S_{i,j}$ — вираз для джерела.

Після перетворення рівняння (2.96) в систему лінійних алгебраїчних рівнянь та її розв'язання, отримуємо розподіл тиску в обчислювальній області. Після цього коректуємо значення поля швидкостей:

$$\boldsymbol{u} = \tilde{\boldsymbol{u}} - \frac{\Delta t}{\rho_{i,j}}\left(\frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x} + \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y}\right) \tag{2.97}$$

Таким чином, отримуємо поле швидкостей на новому часовому кроці. Вирішення рівняння (2.96) зводиться до моделювання розв'язку крайової задачі на основі рівняння Пуассона. Ефективне рішення такої крайової задачі є актуальною проблемою, і подальший матеріал цього розділу буде присвячений розробленому методу моделювання розв'язку крайової задачі на основі рівняння Пуассона для тиску, значення якого використовуються для корекції поля швидкості в методі LBM, за умови нестисливості досліджуваної рідини.

2.2 Висновки до розділу 2

У даному розділі були розглянуті теоретичне обгрунтування можливості застосування методу LBM для моделювання руху рідин та зв'язок між рівнянням Больцмана на рівнянням Нав'є-Стокса, що був встановлений за допомогою розкладу Чепмена-Енського. В контексті розглянутих трьох рівнів абстракції опису суцільного середовища рідини, метод LBM займає місце між мікроскопічним та макроскопічним рівнями - на мезоскопічному рівні, оскільки даний метод оперує усередненими параметрами частинок, що знаходяться в певному об'ємі простору, який називається коміркою. В кожній комірці частинки групуються по дискретним швидкостям, та формують. Дискретні швидкості визначаються певним набором, або решіткою, який позначається *DnQm*. Підбір тої чи іншої решітки швидкостей забезпечує певні властивості рідини, а також визначає наскільки багато необхідно обчислень для моделювання тієї чи іншої схеми. Ми обрали решітку *D2Q9* через

простоту в реалізації, менший об’єм обчислень, в порівнянні зі складнішими схемами та прийнятну точність.

Частинки переміщаються по коміркам у просторі, що представлений решіткою, і їх моделювання зіткнення частинок вимагає відповідну модель. Для цього була використана модель зіткнень Бхатнагара-Гроса-Крука, що приводить функції розподілу частинок до стану локальної рівноваги.

Початкові та граничні умови є необхідною складовою для моделювання фізичних процесів, в основі яких лежать диференціальні рівняння в часткових похідних. Були описані механізми задання початкових значень макроскопічної швидкості та густини, та відповідні формули обрахунку значень функцій розподілу. Були описані граничні умови притоку та витоку рідини та гранична умова твердої стіни, яка використовується для моделювання взаємодії рідини та твердого тіла.

Для досягнення безумовної лінійної стабільності моделювання була описана модифікована функція розподілу на основі мінімізації дискретної ентропії.

Оскільки в основі методів LBM є припущення, що моделюється стислива рідина, виникає необхідність коректування похибки, що виникає при коливанні густини. Тому, було обгрунтовано необхідність уточнення поля швидкості за допомогою рівняння Пуассона для тиску під час моделювання руху нестисливих рідин методом LBM, при малих значеннях числа Маха. Сформульована крайова задача на основі рівняння Пуассона для тиску.

РОЗДІЛ 3
ОСОБЛИВОСТІ МОДЕЛЮВАННЯ РОЗВ’ЯЗКУ КРАЙОВОЇ ЗАДАЧІ НА ОСНОВІ РІВНЯННЯ ПУАССОНА ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Використання нейронних мереж для моделювання розв’язку рівняння Пуассона в модифікованому методі LBM для моделювання нестисливих рідин може мати кілька переваг. Ось кілька аргументів, що обґрунтовують таке використання:

1. Рівняння Пуассона є лінійним диференціальним рівнянням другого порядку. Такі типи рівнянь можуть бути ефективно наближені та розв'язані нейронними мережами, особливо за допомогою глибокого навчання. Особливістю нейронних мереж є самостійне встановлення складних залежностей між вхідними та вихідними даними, що дозволяє їм ефективно наближати функцію розв'язку рівняння Пуассона.
2. Використання нейронних мереж може прискорити обчислення моделювання розв’язку рівняння Пуассона порівняно з традиційними числовими методами, такими як ітераційні методи або методи скінченних різниць. Нейронні мережі можуть працювати паралельно та використовувати оптимізовану обчислювальну архітектуру, таку як графічні процесори (GPU) або спеціалізовані обчислювальні пристрої (наприклад, тензорні процесори), для ефективного обчислення значень тиску.
3. Нейронні мережі можна легко модифікувати та адаптувати для різних початкових та граничних умов. Нейронні мережі можуть бути натреновані на великій кількості даних та потім використовуватись для прогнозування значень тиску в будь-якій точці простору, враховуючи вхідні параметри та граничні умови. Це дозволяє зручно моделювати різні сценарії та випадки без потреби повторного моделювання всієї системи.

Загалом, використання нейронних мереж для моделювання розв’язку рівняння Пуассона в модифікованому методі LBM може бути ефективним підходом, який поєднує швидкість обчислень з гнучкістю та адаптивністю моделі.

Перша частина розділу присвячена формалізації крайової задачі на основі рівняння Пуассона для тиску, який використовується для корекції швидкості при моделюванні нестисливої рідини методом LBM. Друга частина розділу присвячена ітераційним чисельним методам для моделювання розв’язку рівняння Пуассона, які були використані при генерації тренувального та тестового датасетів для розробленої нейронної мережі. Третя частина розділу присвячена особливостям застосування нейронних мереж для моделювання розв’язку диференційних рівнянь в часткових похідних, в тому числі рівняння Пуассона, видам структурних блоків нейронних шарів та функціям активацій, їхнім призначенням та особливостям виконуваних обчислень.

3.1 Крайова задача на основі рівняння Пуассона

Розглянемо більш детально рівняння (2.96). Нехай обчислювальна область дискретизована на сітку, розміром $N \times N$. Тоді $\frac{1}{\rho_{\frac{i+1}{2},j}} = A_{i,j}, \frac{1}{\rho_{\frac{i-1}{2},j}} = B_{i,j}, \frac{1}{\rho_{i,\frac{j+1}{2}}} = C_{i,j}, \frac{1}{\rho_{i,\frac{j-1}{2}}} = D_{i,j}, i = 1,2 \dots N, j = 1,2 \dots N$. Перетворимо рівняння (2.96) у систему лінійних алгебраїчних рівнянь розміру $N^2 \times N^2$ вигляду:

$$\boldsymbol{Au} = \boldsymbol{b} \tag{3.1}$$

де:

$$\boldsymbol{A} = \begin{bmatrix} D_1 & I_{A1} & 0 & 0 & 0 & \cdots & 0 \\ I_{B2} & D_2 & I_{A2} & 0 & 0 & \cdots & 0 \\ 0 & I_{B3} & D_3 & I_{A3} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I_{BN-2} & D_{N-2} & I_{AN-2} & 0 \\ 0 & \cdots & \cdots & 0 & I_{BN-1} & D_{N-1} & I_{AN-1} \\ 0 & \cdots & \cdots & \cdots & 0 & I_{BN} & D_N \end{bmatrix} \tag{3.2}$$

$$D_k = \begin{bmatrix} G_{1,k} & -C_{1,k} & 0 & 0 & 0 & \cdots & 0 \\ -D_{2,k} & G_{2,k} & -C_{2,k} & 0 & 0 & \cdots & 0 \\ 0 & -D_{3,k} & G_{3,k} & -C_{3,k} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -D_{N-2,k} & G_{N-2,k} & -C_{N-2,k} & 0 \\ 0 & \cdots & \cdots & 0 & -D_{N-1,k} & G_{N-1,k} & -C_{N-1,k} \\ 0 & \cdots & \cdots & \cdots & 0 & -D_{N-1,k} & G_{N,k} \end{bmatrix} \quad (3.3)$$

$$I_{Ak} = \begin{bmatrix} -A_{1,k} & 0 & 0 & \cdots & 0 \\ 0 & -A_{2,k} & 0 & \cdots & 0 \\ 0 & 0 & -A_{3,k} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -A_{N,k} \end{bmatrix} \quad (3.4)$$

$$I_{Bk} = \begin{bmatrix} -B_{1,k} & 0 & 0 & \cdots & 0 \\ 0 & -B_{2,k} & 0 & \cdots & 0 \\ 0 & 0 & -B_{3,k} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -B_{N,k} \end{bmatrix} \quad (3.5)$$

$$\boldsymbol{u} = [p_{1,1} \quad p_{2,1} \quad \dots \quad p_{N,1} \quad p_{1,2} \quad p_{2,2} \quad \dots \quad p_{N,2} \quad \dots \quad p_{N,N}]^{\mathbf{T}} \quad (3.6)$$

$$\boldsymbol{b} = [-S_{1,1} \quad -S_{2,1} \quad \dots \quad -S_{N,1} \quad -S_{1,2} \quad -S_{2,2} \quad \dots \quad -S_{N,2} \quad \dots \quad -S_{N,N}]^{\mathbf{T}} \quad (3.7)$$

де $k = 1,2, \dots, N$.

Існує чимало методів розв’язання отриманої системи рівнянь, огляд яких було зроблено в розділі 1.2. Ми розглянемо ітераційні чисельні методи розв’язання систем лінійних алгебраїчних рівнянь. Вони були використані під час підготовки даних для розробленого нами методу, також з ними були здійснене порівняння у швидкодії нашого методу.

3.2 Ітераційні чисельні методи

Ітераційні чисельні методи є одним з основних підходів до розв'язання систем лінійних алгебраїчних рівнянь (СЛАР). Замість прямого розв'язання СЛАР за допомогою прямих методів, ітераційні методи розв'язують систему крок за кроком, наближаючи справжнє розв'язання через послідовні ітерації.

Одним з найвідоміших ітераційних методів є метод Якобі. У методі Якобі система лінійних рівнянь розбивається на сукупність рівнянь, і кожне рівняння

використовується для оновлення значень невідомих з попередньої ітерації. Ітераційний процес продовжується до досягнення заздалегідь встановленої точності.

Інший популярний метод - метод Гауса-Зейделя - також розбиває систему на сукупність рівнянь, але в цьому випадку кожне рівняння використовує найновіші значення невідомих з поточної ітерації. Це призводить до швидшої збіжності порівняно з методом Якобі.

Метод релаксації (англ. successive over-relaxation, SOR) є модифікованим методом Гауса-Зейделя. Відзначено що він сходиться в багатьох випадках швидше, ніж інші ітераційні методи.

Розглянемо згадані вище ітераційні методи більш детальніше.

3.2.1 Метод Якобі

Метод Якобі — це ітераційний чисельний метод, який використовується для розв'язування систем лінійних рівнянь. Названо на честь німецького математика Карла Густава Якоба Якобі. Метод особливо корисний для великих розріджених систем, де більшість коефіцієнтів у матриці, що представляє систему, дорівнює нулю.

Метою методу Якобі є ітераційне уточнення початкового припущення рішення, доки воно не зійдеться до точного рішення. Метод заснований на розбитті вихідної системи рівнянь на діагональну складову і недіагональну складову.

Розглянемо систему лінійних рівнянь у матричній формі: $\boldsymbol{Ax} = \boldsymbol{b}$, де $\boldsymbol{A}$— матриця коефіцієнтів, $\boldsymbol{x}$— вектор невідомих, $\boldsymbol{b}$ — вектор вільних членів.

Метод Якобі починається з перестановки рівняння, щоб виділити кожну невідому змінну в лівій частині рівняння:

$$x_1 = \frac{\left(b_1 - (A_{12}\,{}^{\star}x_2 + A_{13}\,{}^{\star}x_3 + \cdots + A_{1n}\,{}^{\star}x_n)\right)}{A_{11}}$$

$$x_2 = \frac{\left(b_2 - (A_{21}\,{}^{\star}x_1 + A_{23}\,{}^{\star}x_3 + \cdots + A_{2n}\,{}^{\star}x_n)\right)}{A_{22}} \qquad (3.8)$$

…

$$x_n = \frac{\left(b_n - \left(A_{n1}{}^{*}x_1 + A_{n2}{}^{*}x_2 + \cdots + A_{nn-1}{}^{*}x_{n-1}\right)\right)}{A_{nn}}$$

Тут кожне рівняння представляє правило оновлення для кожної невідомої змінної. Ітерація починається з початкового припущення для $\boldsymbol{x}$, і в кожній ітерації значення $\boldsymbol{x}$ оновлюються відповідно до наведених вище рівнянь. Процес триває до тих пір, поки не буде виконано критерій збіжності, наприклад, досягнуто заданої точності або максимальної кількості ітерацій.

Метод Якобі вимагає, щоб матриця коефіцієнтів $\boldsymbol{A}$ була діагонально домінантною або симетричною позитивно визначеною для збіжності. Якщо матриця не задовільняє цим умовам, метод може не сходитися або сходитися повільно. Дана умова виконується для матриці $\boldsymbol{A}$ з формули (3.1).

Але даний метод, як й інші ітераційні методи, для виконання умови збіжності вимагає значну кількість операцій. Тому використання іншого методу розв’язання рівняння (2.96) є актуальним. Перспективним кандидатом є метод заснований на машинному навчанні та штучних нейронних мережах, оскільки час обчислення нейронної мережі є фіксованим, використання графічних прискорювачів дозволяє досягти задовільної швидкості обчислень. Далі опишемо розроблену нейронну мережу для моделювання розв’язку крайової задачі на основі рівняння Пуассона, особливості її тренування.

3.2.2 Метод Гауса-Зейделя

Метод Гауса-Зейделя — це ітераційний чисельний метод, який використовується для розв’язування систем лінійних рівнянь. Він названий на честь німецьких математиків Карла Фрідріха Гауса та Філіпа Людвіга фон Зайделя. Цей метод є вдосконаленням у порівнянні з методом Якобі та забезпечує швидшу конвергенцію для багатьох типів систем.

Враховуючи систему лінійних рівнянь, представлену як $\boldsymbol{Ax} = \boldsymbol{b}$, де $\boldsymbol{A}$ — матриця коефіцієнтів $N \times N$, $\boldsymbol{x}$ — вектор розв’язку, а $\boldsymbol{b}$ — вектор правої частини,

метод Гаусса-Зайделя ітеративно оновлює компоненти розв'язку вектор із використанням найновіших значень.

Алгоритм роботи методу наступний:

1. Початкове припущення для вектора розв'язку $\boldsymbol{x}^{(\mathbf{0})}$
2. Для кожного рівняння в системі оновіть компоненти вектора розв'язку, використовуючи найновіші значення. Для i-го рівняння формула оновлення має вигляд:

$$x_i^{(k+1)} = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij} x_j^{(k)} \right),\ i = 1,2,\dots,N \qquad (3.9)$$

де $x_i^{(k+1)}$— оновлене значення i-го компонента вектора розв'язку в $(k+1)$-й ітерації, b_i — i-й компонент правого вектора, A_{ij} є коефіцієнтом i-го рівняння в j-му стовпці матриці A, а $x_j^{(k)}$ є j-м компонентом вектора розв'язку на k-й ітерації. Метод Гауса-Зейделя використовує найновіші значення для $x_j^{(k+1)}$ коли обчислює оновлене значення $x_i^{(k+1)}$

3. Повторити крок 2, поки рішення не збіжиться, тобто поки зміна вектора рішення між двома послідовними ітераціями не впаде нижче заданого порогу або не буде досягнуто максимальної кількості ітерацій.

Метод Гаусса-Зайделя покращує збіжність порівняно з методом Якобі, використовуючи найновіші значення, доступні під час процесу ітерації. Однак важливо зазначити, що метод може не збігатися для всіх типів матриць, особливо якщо матриця є погано обумовленою або не задовольняє певним властивостям, таким як діагональне домінування.

3.2.3 Метод релаксацій

Метод релаксацій — це ітераційний чисельний метод, який використовується для розв'язування систем лінійних рівнянь. Це розширення методу Гауса-Зейделя та включає додатковий параметр релаксації для прискорення збіжності.

Алгоритм роботи методу релаксацій наступний:

1. Початкове припущення для вектора розв'язку $\boldsymbol{x}^{(\mathbf{0})}$

2. Вибрати такий параметр релаксації ω, щоб $0 < \omega < 2$. Величина ω впливає на швидкість збіжності методу.

3. Для кожного рівняння в системі оновити компоненти вектора розв'язку за допомогою формули:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij}x_j^{(k)}\right),\ i = 1,2\ldots,N \quad (3.10)$$

4. Повторити крок 3, доки розв'язок не збіжиться.

Метод SOR прискорює збіжність шляхом введення параметра релаксації ω. Вибираючи оптимальне значення для ω, збіжність може бути досягнута швидше, ніж метод Гауса-Зейделя. Оптимальне значення ω залежить від конкретної проблеми та може бути визначено шляхом експерименту або теоретичного аналізу.

Варто зазначити, що якщо $\omega = 1$, метод SOR зводиться до методу Гауса-Зейделя.

Використання відповідного значення параметра релаксації ω має вирішальне значення для успіху методу релаксацій. Значення ω більше 1 можуть призвести до швидшої конвергенції (надмірна релаксація), а значення менше 1 можуть уповільнити конвергенцію (недостатня релаксація).

Але даний метод, як й інші ітераційні методи, для виконання умови збіжності вимагає значну кількість операцій. Тому використання іншого методу розв'язання рівняння (2.96) є актуальним. Перспективним кандидатом є метод заснований на машинному навчанні та штучних нейронних мережах, оскільки час обчислення нейронної мережі є фіксованим, використання графічних прискорювачів дозволяє досягти задовільної швидкості обчислень. Далі розглянемо особливості застосування нейронних мереж для моделювання розв'язку крайової задачі на основі рівняння Пуассона.

3.3 Особливості застосування нейронних мереж для моделювання розв’язку крайової задачі на основі рівняння Пуассона

Нейронні мережі є одним із багатьох алгоритмів машинного навчання, який дозволяє моделювати складні залежності і здійснювати прогнозування на основі великого обсягу даних. Вони натхненні біологічною нервовою системою і здатні автоматично вивчати та адаптуватися до вхідних даних без явного програмування [203].

Нейронні мережі складаються зі з'єднаних штучних нейронів, які функціонують разом для обробки вхідних даних та генерації вихідних результатів. Кожен штучний нейрон отримує вхідні дані, обчислює зважену суму цих даних та застосовує активаційну функцію для генерації вихідного сигналу. Штучні нейрони групуються в шари, інтерпретуючи інформацію з одного шару в інший.

3.3.1 Формалізація задачі моделювання розв’язку крайової задачі на основі рівняння Пуассона за допомогою штучної нейронної мережі

Моделювання розв’язку крайової задачі на основі рівняння Пуассона за допомогою штучної нейронної мережі можна звести до задачі регресії у статистичному моделюванні та машинному навчанні [204], оскільки результатом вирішення задачі є значення поля тиску на дискретизованій обчислювальній області.

Задача регресії в контексті машинного навчання – це прогнозування неперервної величини, яка може варіюватись у широкому діапазоні значень.

Нехай X-множина об’єктів вхідних значень, або ознак, де $X = \{X_1, \dots, X_N\}, X_i \in \mathbb{R}^{M_X}$, M_X – розмірність простору ознак. Тоді задачу регресії можна привести до функції R, яка на основі певної множини параметрів $W = \{w_1, \dots, w_{M_W}\}, w_i \in \mathbb{R}$ буде передбачувати цільове значення $\widetilde{Y}_i, \widetilde{Y}_i \in \mathbb{R}^{M_Y}$, для вхідного об'єкту ознак X_i:

$$R(W, X_i): X_i \to \widetilde{Y}_i \tag{3.11}$$

Параметри функції регресії W визначаються за допомогою мінімізації функції втрат, яка визначає, наскільки точно функція R передбачає цільову змінну:

$$L(R(W,X),Y) \to \min_{W} \tag{3.12}$$

Де $Y = \{Y_1, \dots, Y_N\}, Y_i \in \mathbb{R}^{M_Y}$ – множина цільових значень для множини об'єктів X.

Для задачі регресії функціями втрат, які найчастіше використовуються у роботах присвячених застосуванню нейронних мереж, є наступні:

1. **Середньоквадратична помилка** : середньоквадратична помилка є однією з найпоширеніших функцій втрат у регресії. Він обчислює середню квадратичну різницю між спрогнозованими значеннями і справжніми значеннями цільової змінної:

$$L(\tilde{Y}, Y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}\left(Y_i - \widetilde{Y}_\iota\right)^2 \tag{3.13}$$

2. **Середньоабсолютна помилка** : Середньоабсолютна помилка вимірює середню абсолютну різницю між прогнозованими та справжніми значеннями:

$$L(\tilde{Y}, Y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}\left|Y_i - \widetilde{Y}_\iota\right| \tag{3.14}$$

3. **Функція втрат Хьюбера:** це гібридна функція втрат, яка поєднує в собі характеристики попередніх двох описаних функцій. Вона менш чутлива до викидів порівняно з середньоквадратичною функцією і забезпечує плавний градієнт, як середньоабсолютна помилка. Функція втрат для однієї пари Y_i та $\widetilde{Y}_\iota$ виглядає наступним чином:

$$L_H\left(\widetilde{Y}_\iota, Y_i\right) = \begin{cases} \frac{1}{2}a^2 & |a| \le \delta \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & |a| > \delta \end{cases} \tag{3.15}$$

де $a = Y_i - \widetilde{Y}_\iota$, δ – параметр, який поріг для перемикання між квадратичною і лінійною поведінкою. Тоді повна функція втрат виглядає так:

$$L(\tilde{Y}, Y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} L_H\left(\widetilde{Y}_\iota, Y_i\right) \tag{3.16}$$

3.3.2 Шари штучних нейронних мереж

Структура нейронних мереж складається з блоків, які виконують певний вид обчислень над вхідними даними, та передають вихідні дані на наступний блок або на вихід нейронної мережі. Ці блоки називаються шарами нейронних мереж. Типів цих шарів є велика кількість, тому далі ми опишемо тільки ті, які були використані в розробленій нами нейронній мережі

Згортковий шар

Згортковий шар складається з набору доступних для вивчення фільтрів або ядер. Кожен фільтр — це невелика матриця ваг, яка застосовується до всіх вхідних даних для обчислення вихідних даних шару. Фільтр переміщується над вхідними даними у вигляді ковзного вікна, виконуючи поелементне множення між вагами фільтра та відповідними вхідними значеннями, а потім підсумовуючи результати для створення єдиного значення на вихідному тензорі.

Ключова ідея згорткового рівня полягає в тому, щоб охопити локальні моделі або просторові ієрархії, присутні у вхідних даних. Пересуваючи фільтри по вхідних даних, шар може виявляти різні елементи, такі як краї, кути або текстури в різних просторових місцях. Ваги фільтрів вивчаються під час процесу навчання, що дозволяє згортковому шару адаптуватися та автоматично розпізнавати відповідні шаблони.

Крім того, згортковий рівень часто включає нелінійні функції активації, після кожної поелементної суми. Функція активації вводить нелінійність у шар, дозволяючи моделі фіксувати складніші зв’язки між функціями. Функції активації розглянуті детальніше в розділі 3.3.3.

Припустимо, що вхідний зразок даних X_i – це тензор розміру $W_1 \times H_1 \times D_1$. Далі згортковий шар, який приймає такі вхідні дані, має наступні параметри: кількість фільтрів K, розмір фільтру F (тоді кожен фільтр являє собою тензор розміру $F \times F \times D_1$), крок (англ. stride) S, з яким фільтр рухається по вхідним даним, доповнення (англ. padding) P – визначає кількість нульових рядків і стовпців, які будуть додані до кожної матриці розміру $W_1 \times H_1$, з яких складається

X_i. Параметр P дозволяє керувати розміром вихідного тензору. Це є особливо корисно, коли необхідно, щоб розмір вихідного тензору був ідентичний вхідному розміру. Відповідно, згортковий шар обчислює результуючий тензор розміру $W_2 \times H_2 \times D_2$, де $W_2 = (W_1 - F + 2P)/S + 1$, $H_2 = (H_1 - F + 2P)/S + 1$, $D_2 = K$. Візуалізація роботи згорткового шару, при параметрах $W_1 = 6, H_1 = 6, D_1 = 3, K = 1, F = 3, S = 1, P = 0$ зображена на рис.3.1.

Шар пакетної нормалізації

Пакетна нормалізація — це техніка, яка використовується в нейронних мережах, зокрема в моделях глибокого навчання, для вирішення проблеми внутрішнього зсуву коваріантних і прискорення конвергенції навчання [205]. Шар пакетної нормалізації вставляється між згортковими або повністю з'єднаними шарами нейронної мережі.

Внутрішній коваріативний зсув відноситься до зміни в розподілі мережевих активацій під час навчання. Це може перешкоджати процесу навчання, оскільки кожен рівень повинен постійно адаптуватися до мінливого розподілу вхідних даних. Шар пакетної нормалізації вирішує цю проблему шляхом нормалізації проміжних активацій рівня нейронної мережі в міні-пакетах даних.

Шар пакетної нормалізації працює таким чином:

1. Під час навчання рівень нормалізації пакету даних обчислює такі статистичні значення як середнє значення та стандартне відхилення активацій у кожній міні-серії. Це забезпечує оцінку статистики міні-пакету даних.

2. Після цього, активації в кожному міні-пакеті даних нормалізуються за допомогою середнього значення та стандартного відхилення, обчисленого на попередньому кроці. Це робиться для центрування та масштабування активацій.

3. Наступним кроком є зсув та масштабування нормалізованих активацій за допомогою параметрів, які можна вивчати, відомих як гамма та бета. Ці параметри дозволяють шару пакетної нормалізації вивчати оптимальний масштаб і зсув для кожної активації, надаючи мережі гнучкісті.

4. Під час кроку зворотного розповсюдження обчислюються градієнти для гамма- та бета-параметрів, що дозволяє шару пакетної нормалізації вивчати відповідне масштабування та зміщення для активацій.

Переваги пакетної нормалізації наступні:

1. Нормалізуючи активації, пакетна нормалізація допомагає стабілізувати процес навчання та прискорює збіжність. Це зменшує залежність кожного шару від зміни розподілу активацій попередніх шарів.

2. Пакетна нормалізація діє як форма регулярізації, додаючи шум до мережі. Цей шум допомагає запобігти перенавчанню та може зменшити потребу в інших методах регуляризації.

3. Нормалізація активацій дозволяє швидше тренувати нейронну мережу без ризику відхилень або коливань під час навчання. Це забезпечує швидшу збіжність та зменшує чутливість до вибору гіперпараметрів навчання.

4. Було виявлено, що пакетна нормалізація покращує здатність нейронних мереж до узагальнення, дозволяючи їм краще працювати з невидимими даними.

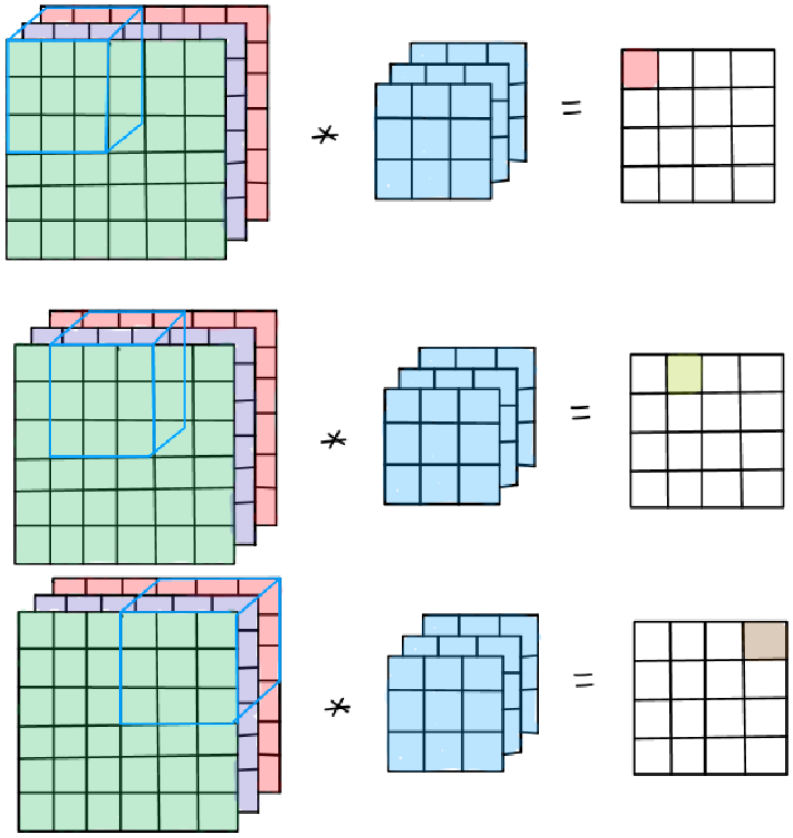

Рисунок 3.1 – Зображення роботи згорткового шару

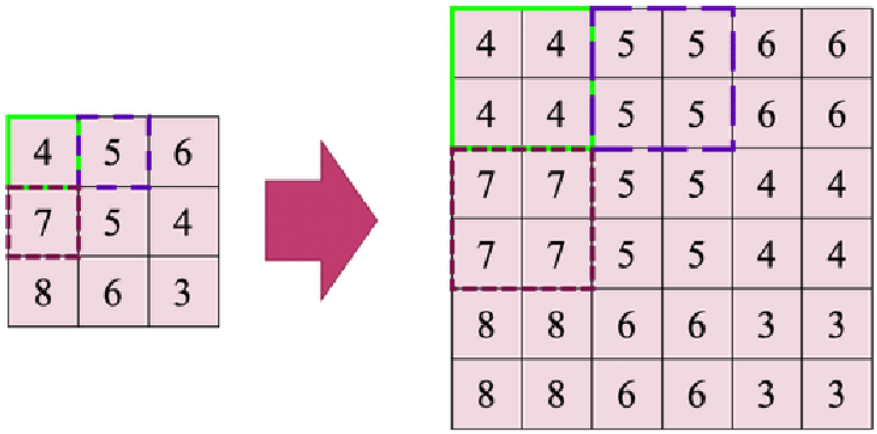

Рисунок 3.2 – Візуалізація роботи шару збільшування роздільності

3.3.3 Функції активації

Функції активації відіграють вирішальну роль у штучних нейронних мережах, вносячи нелінійність у вихід окремих нейронів або шарів. Вони визначають, чи має бути активований нейрон, на основі зваженої суми його вхідних даних. Функції активації дозволяють нейронним мережам моделювати складні нелінійні зв’язки між входом і виходом, що робить їх здатними вирішувати широкий спектр завдань [206].

Основні ролі та переваги функцій активації в штучних нейронних мережах такі:

- Функції активації вводять нелінійність у мережу, дозволяючи їй вивчати та представляти складні моделі та зв’язки в даних. Без нелінійних функцій активації нейронна мережа була б обмежена вивченням лише лінійних зв’язків, що суттєво обмежує її ефективність.
- За допомогою функцій активацій можна визначати діапазон чисельних значень, які будуть на виході нейрону або шару. Обмежуючи вихідний діапазон, функції активації можуть нормалізувати або масштабувати вихідні значення, гарантуючи, що вихідні дані мережі знаходяться в певному бажаному діапазоні або мають певний розподіл.
- Функції активації допомагають поширювати сигнали в мережі. Вони визначають рівень активації нейронів на основі отриманих вхідних даних,

дозволяючи мережі передавати та обробляти інформацію з одного рівня на інший. Функції активації забезпечують механізм для сильної активації нейронів або залишаються здебільшого неактивними залежно від величини та характеру вхідних даних.

- Певні функції активації, такі як зрізаний лінійний вузол (англ. Rectified Linear Unit ReLU) та її варіанти, мають властивості регуляризації. Ці функції активації вводять розрідженість шляхом обнулення від'ємних значень, що може допомогти запобігти перенавчанню та покращити здатність мережі до узагальнення.
- Функції активації спрощують інтерпретацію та розуміння поведінки мережі. Досліджуючи патерни активації, можна зрозуміти, які нейрони активуються для різних вхідних стимулів, допомагаючи в аналізі та інтерпретації моделі.

Основними функціями активації, які використовуються в задачах регресії є наступні:

- ReLU

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{3.17}$$

- PReLU (Parametric rectified linear unit)

$$f(\alpha, x) = \begin{cases} \alpha x & x < 0 \\ x & x \geqslant 0 \end{cases} \tag{3.18}$$

- ELU

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & x \leq 0 \\ x & x > 0 \end{cases} \tag{3.19}$$

- Softplus

$$f(x) = \ln(1 + e^x) \tag{3.20}$$

- Сігмоїда

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.21}$$

- Гіперболічний тангенс

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{3.22}$$

3.3.4 Загальний процес моделювання розв’язку диференціальних рівнянь за допомогою штучних нейронних мереж

Моделювання розв’язку диференціальних рівнянь із частинними похідними за допомогою штучних нейронних мереж є новим напрямком. Штучні нейронні мережі, включаючи багатошарові персептрони і згорткові нейронні мережі, пропонують підхід на основі даних для наближення розв’язків диференціальних рівнянь з частинними похідними безпосередньо з вхідних даних.

Загальна ідея використання штучних нейронних мереж для диференціальних рівнянь полягає в тому, щоб навчити нейронну мережу вивчати основні закономірності та зв’язки між вхідними даними (просторовими та часовими координатами) і рішенням диференціального рівняння. Використовуючи великі набори даних і потужні обчислювальні ресурси, нейронні мережі можуть апроксимувати складні рішення диференціальних рівнянь більш ефективно, ніж традиційні аналітичні або чисельні методи.

Ось загальний огляд процесу вирішення диференціальних рівнянь за допомогою штучних нейронних мереж:

1. Формулювання задачі, що включає в себе рівняння, граничні умови та початкові умови (якщо є). Диференціальне рівняння повинно мати унікальне рішення.
2. Генерація навчальних даних, за допомогою моделювання розв’язку диференціального рівняння для різних початкових та вхідних умов. Це можна зробити за допомогою чисельних методів, аналітичних методів або за допомогою існуючих рішень. Набір даних має складатися з пар входів і виходів, де входами є просторові координати та/або час, а виходами є відповідні рішення.
3. Проектування архітектури штучної нейронної мережі. Для цього спочатку необхідно обрати відповідну архітектуру нейронної мережі, яка підходить для проблеми. Наприклад, багатошаровий персептрон можна використовувати для загальних проблем диференціальних рівнянь, тоді як

згорткові нейронні мережі ефективні для проблем із просторовими структурами, такими як рівняння на основі зображень. Архітектура включає кількість шарів, кількість нейронів у кожному шарі, функції активації та будь-які додаткові шари чи операції, характерні для проблеми.

4. Визначення відповідної функції втрат, яка кількісно визначає розбіжність між прогнозованим рішенням і істинним рішенням. Загальні варіанти включають середню квадратичну помилку (MSE) або середню абсолютну помилку (MAE).

5. Навчання штучної мережі за допомогою згенерованого набору даних шляхом оптимізації функції втрат. Це передбачає коригування вагових коефіцієнтів і зміщень нейронної мережі за допомогою алгоритмів оптимізації (наприклад Adam або стохастичний градієнтний спуск). Процес навчання ітеративно оновлює параметри мережі, щоб мінімізувати втрати та підвищити точність прогнозів.

6. Оцінка точності моделі після навчання для нових вхідних умов. Навчена штучна нейронна мережа діє як швидкий і ефективний розв’язувач для диференціальних рівнянь з частинними похідними, надаючи рішення з прийнятною точністю.

3.4 Висновки до розділу 3

У даному розділі дисертації було досліджено особливості використання нейронних мереж для моделювання розв’язку рівняння Пуассона.

Використання нейронних мереж для моделювання розв’язку рівняння Пуассона є багатообіцяючим підходом. Нейронні мережі можуть задавати набагато складніші функції джерела, ніж традиційні аналітичні або чисельні методи, тому вони можуть надати не менш точні результати при моделювання розв’язку крайових задач.

Були розглянуті ітераційні чисельні методи вирішення систем алгебраїчних рівнянь, які традиційно використовуються для моделювання розв’язку рівняння

Пуассона. Основні недоліки ітераційних чисельних методів підштовхують до розробки альтернативного методу моделювання розв’язку рівняння Пуассона, на основі нейронних мереж який буде швидшим і даватиме точні результати. Використання одного із ітераційних чисельних методів є доцільним для генерації навчального датасету для нейронної мережі.

Використання нейронних мереж для моделювання розв’язку крайової задачі на основі рівняння Пуассона має свої особливості. моделювання розв’язку крайової задачі на основі рівняння Пуассона за допомогою штучної нейронної мережі можна зводиться задачі регресії у машинному навчанні, оскільки результатом вирішення задачі є значення поля тиску на дискретизованій обчислювальній області є дійсними числами. В рамках формалізації задачі моделювання розв’язку рівняння Пуассона були розглянуті та описані функції втрат, які мінімізуються в ході вирішення задачі регресії в машинному навчанні.

Були розглянуті різні шари штучних нейронних мереж та функції активацій, які використовуються для досягнення бажаних результатів. При виборі структурних блоків та функцій активацій необхідно враховувати конкретні вимоги задачі та особливості даних.

Багатошаровий персептрон та згорткові нейронні мережі - це два різні типи нейронних мереж, і кожен з них має свої особливості та застосування для моделювання розв’язку рівняння Пуассона та схожих задач. Так, застосування згорткових нейронних мереж є доцільним для даних з локальною структурою. Також Вони можуть автоматично виділяти інформацію про розташування об'єктів у великих даних, яка може бути корисною для моделювання розв’язку задачі Пуассона. Багатошарові персептрони не мають таких переваг, то використання згорткових нейронних мереж є доцільним для моделювання розв’язку рівняння Пуассона.

В кінці розділу приведений загальний огляд процесу вирішення диференціальних рівнянь за допомогою штучних нейронних мереж – формулювання задачі, генерація навчальних даних, проектування архітектури нейронної мережі, вибір відповідної функції втрат, навчання штучної нейронної

мережі та оцінка точності моделі. Цей процес буде використаний при розробці конкретної нейронної мережі для моделювання розв’язку рівняння Пуассона в розділі 4.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ДВОРІВНЕВОГО МЕТОДУ МОДЕЛЮВАННЯ РУХУ РІДИНИ ЗА ДОПОМОГОЮ РЕШІТЧАСТОЇ МОДЕЛІ БОЛЬЦМАНА ТА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

В результаті здійсненого дослідження був запропонований дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі. Розроблений метод включає в собі дві складові:

1. Модифікований метод LBM,
2. Згорткову нейронну мережу для моделювання розв'язку крайової задачі на основі рівняння Пуассона для тиску

Даний метод дозволяє моделювати рух нестисливої рідини більш точно, ніж класичні методи LBM.

4.1 Структура нейронної мережі для моделювання розв'язку крайової задачі на основі рівняння Пуассона

Опишемо структуру розробленої нейронної мережі, яка моделює розв'язок крайової задачі на основі рівняння Пуассона. В якості підходу до побудови нейронної мережі був обраний тип нейронної мережі, під назвою FPN (англ. Feature Pyramid Network) [207].

Структура запропонованої нейронної мережі складається з двох частин: «bottom-up pathway» і «top-down pathway». Повна структура нейронної мережі зображена на рис.2.12. Вхідними даними для частини «bottom-up pathway» є тензор розміру $96 \times 96 \times 2$, який є комбінацією двох матриць розміру 96×96, оскільки в проведених експериментах, які описані в розділі 5, обчислювальна область поділена на квадрат, 96×96 однаковими комірками. Перша матриця – це значення правої частини рівняння (3.1), для обчислення якої використано рівняння (2.95) (f_input на рис.4.4), друга матриця – кодування обчислювального простору ($dist\_input$ на рис. 4.4). Для кодування простору використовується матриця geo

розміру 96×96, де кожен її елемент може мати значення 1, якщо відповідна комірка містить тверде тіло, або 0, якщо там рідини. Далі відповідно вираховується матриця $dist$, кожен елемент $dist_{ij}$ обчислюється наступним чином: якщо $geo_{ij} = 1$, то $dist_{ij} = 0$, в іншому випадку значення $dist_{ij}$ дорівнює відстані до найближчої комірки з твердим тілом, де ширина і сторона комірки дорівнює $\frac{1}{96\sqrt{2}}$. Зображення кодування простору на рис. 4.1. Даний підхід дозволяє використати нейронну мережу при моделюванні руху рідин в обчислювальних областях зі складною довільною геометрією.

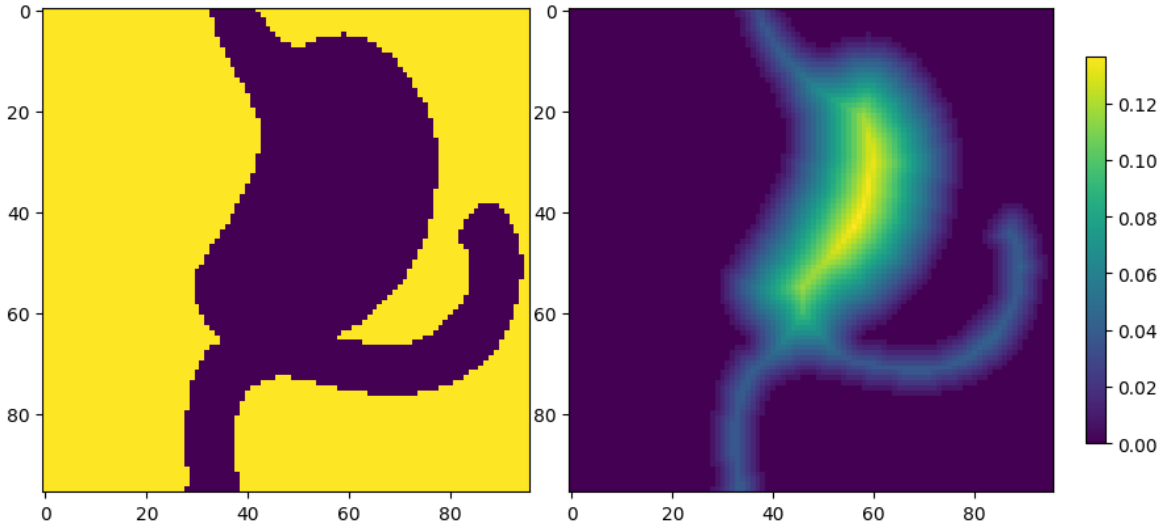

Рисунок 4.1 – Зліва матриця geo, жовтий колір – тверде тіло, фіолетовий колір – рідини; справа – матриця $dist$

Частина «bottom-up pathway» складається з 5 блоків – чотирьох «residual» блоків та одного «pre_conv» блоку. Перший – це «pre_conv» блок, що складається з згорткового шару, з параметрами $K = 64, F = 5, S = 1$, шару пакетної нормалізації та функції активації ReLU. Далі кожен «residual» блок складається із згорткового блоку та двох «identity» блоків. Структури «identity» та згорткового блоків зображені на рис.4.2 та 4.3 відповідно.

Оскільки згорткові та «identity» блоки розташовані послідовно в «bottom-up pathway», тому всі чотири «residual» блоки можна описати за допомогою таблиці 4.1, де міститься вся інформація про кількість та параметри згорткових шарів.

Кожен наступний «residual» блок ділить навпіл тензор, отриманий з попереднього блоку, і передає результат на відповідний шар збільшення

роздільності, який розташований у частині «top-down pathway». Запропонована структура згорткової нейронної мережі використовує основну властивість нейронних мереж цього типу, яка поєднує в собі характеристики рідини в різних масштабах. Таким чином, ми поєднуємо кожну комірку обчислювальної області зі всією областю. Вихідні дані кожного «residual» блоку та блоку «pre_conv» передаються на згортковий шар з параметрами $K = 32, F = 1, S = 1$, щоб зменшити кількість каналів.

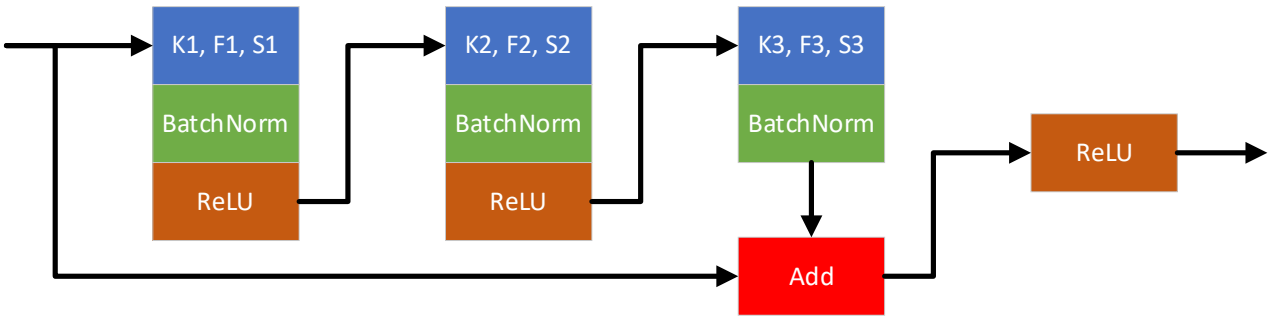

Рисунок 4.2 – Структура «identity» блоку. Сині блоки – згорткові шари з відповідним параметрами. Блок «add» здійснює додавання тензорів однакового розміру

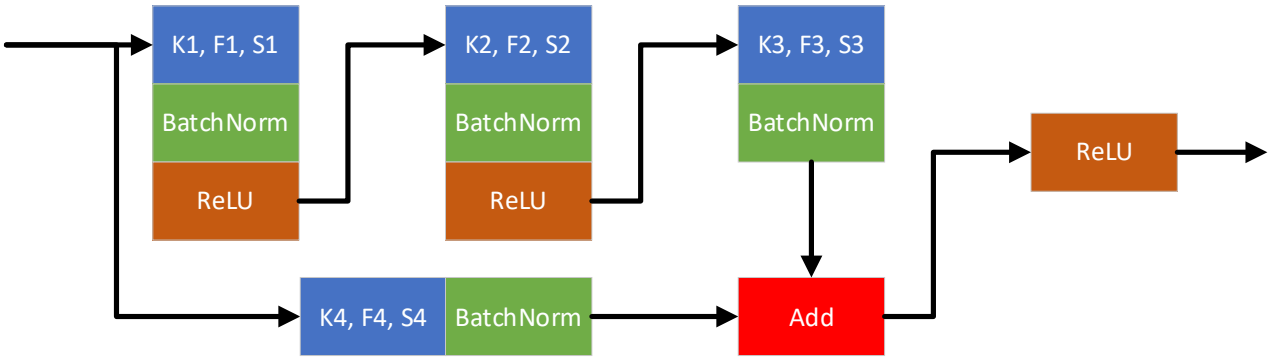

Рисунок 4.3 – Структура згорткового блоку

Далі вихідні дані згорткових шарів, що зменшили розмірність вихідних даних «residual» блоків йдуть на вхід до шарів збільшення роздільності, які формують «top-down pathway», де розмір виходу першого «residual» блоку множиться на 2, другого виходу на 4, а третього виходу на 8 і четвертого на 16. Після цієї операції всі виходи об'єднуються в один тензор, шляхом їх компонування уздовж останньої осі. Нарешті, цей тензор переходить до двовимірного блоку «output_conv», який складається з 4 шарів згортки, кожен з яких має розмір ядра $F = 3$: перший шар складається з 32 фільтрів, другий містить

16 фільтрів, третій – 8 фільтрів і четвертий має один фільтр та функції активації гіперболічного тангенса в діапазоні від -1 до 1. Дана функція активації була обрана через особливості тренування, які будуть описані в розділі 4.1.1, та властивості системи лінійних алгебраїчних рівнянь, до якої зводиться вирішувана крайова задача. На вхід кожного із згорткових шарів блоку «output_conv» подано додатково новий вхідний тензор «rho_input» розміром 96×96, в якому містяться значення поля густини.

Таблиця 4.1 – Структура «residual» блоків «bottom-up pathway» частини нейронної мережі.

Номер "residual" блоку	Підблок	Параметри
1	Згортковий	K1=32, F1=1, S1=2, K2=32, F2=3, S2=1, K3=64, F3=1, S3=1, K4=64, F4=1, S4=2
	"identity"	K1=32, F1=1, S1=1, K2=32, F2=3, S2=1, K3=64, F3=1, S3=1
	"identity"	K1=32, F1=1, S1=1, K2=32, F2=3, S2=1, K3=64, F3=1, S3=1
2	Згортковий	K1=48, F1=1, S1=2, K2=48, F2=3, S2=1, K3=96, F3=1, S3=1, K4=96, F4=1, S4=2
	"identity"	K1=48, F1=1, S1=1, K2=48, F2=3, S2=1, K3=96, F3=1, S3=1
	"identity"	K1=48, F1=1, S1=1, K2=48, F2=3, S2=1, K3=96, F3=1, S3=1
3	Згортковий	K1=64, F1=1, S1=2, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1, K4=128, F4=1, S4=2
	"identity"	K1=64, F1=1, S1=1, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1
	"identity"	K1=64, F1=1, S1=1, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1
4	Згортковий	K1=64, F1=1, S1=2, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1, K4=128, F4=1, S4=2
	"identity"	K1=64, F1=1, S1=1, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1
	"identity"	K1=64, F1=1, S1=1, K2=64, F2=3, S2=1, K3=128, F3=1, S3=1

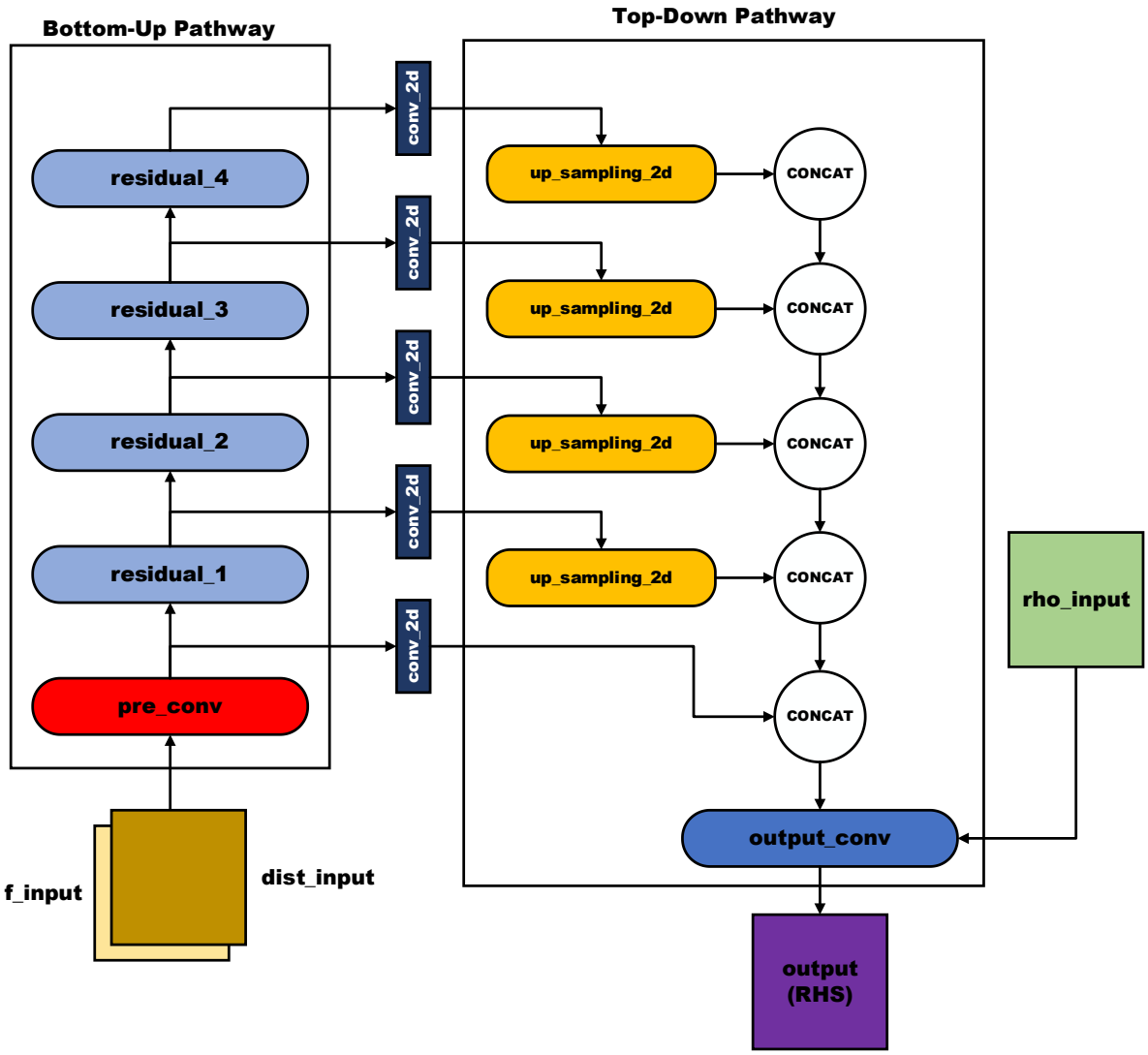

Рисунок 4.4 – Загальна структура розробленої нейронної мережі

Вихідними даними нейронної мережі є двовимірний масив розміром 96×96. Це не є фінальним розв'язком крайової задачі. Отримання фінального розв'язку описано в розділі 4.1.1.

4.1.1 Особливості генерування тренувальних даних для нейронної мережі

Стабільна та точна робота нейронної мережі залежить від якості тренувальних даних. Моделі глибокого навчання спрямовані на вивчення шаблонів і представлень даних, щоб робити прогнози. Високоякісний навчальний набір даних повинен охоплювати різноманітну та репрезентативну вибірку мінливості та складності реального світу, з якою очікується зіткнутися модель під чассвоєї

роботи. Це гарантує, що модель може добре узагальнювати невидимі дані та робити точні прогнози в різних сценаріях. Моделі глибокого навчання часто вимагають великих обсягів даних для охоплення складних базових шаблонів. Наявність достатнього обсягу високоякісних даних дозволяє моделі вивчати надійні представлення та добре узагальнювати.

Щоб отримати датасет, який дозволить навчити нейронну мережу для моделювання розв'язку рівняння Пуассона тиску, значення якого будуть використані для корекції швидкості під час моделювання методом LBM, описаним в попередніх розділах, ми використали метод LBM, який моделював рух рідин в різних обчислювальних областях, тобто з різними просторовими конфігураціями твердого тіла та простору, куди могла потрапити рідина, різними параметрами релаксації та різноманітними початковими умовами.

Ми підготували 15 обчислювальних областей, з різними геометріями перешкод, або твердого тіла (рис. 4.1). Для кожної області ми запустили процес моделювання методом LBM при різних випадкових значеннях початкової швидкості рідини, її густини та часу релаксації. Параметр релаксації τ був у діапазоні від 0.1 до 0.6, густина ρ – в межах від 500 до 2000, початкова швидкість рідини – від 0.002 до 0.01. На кожній ітерації, перед розрахунком рівноважної функції на кроці зіткнення, розраховувались значення вільних членів, за допомогою правої частини рівняння (2.95) та матриця $\boldsymbol{A}$ рівняння (3.1). Потім складалась відповідна система лінійних рівнянь, яка розв'язувалась методом Якобі. Розв'язком цієї системи рівнянь є значення тиску в обчислювальній області. Отримані протягом моделювання значення вільних членів та густини зберігались як ознаки для нейронної мережі, а значення тиску – як цільова змінна.

Для забезпечення стабільності тренування нейронної мережі та її точності, значення ознак та цільових змінних необхідно обмежувати в певному діапазоні [210]. Тому в згенерованому датасеті, всі значення повинні лежати в діапазоні [-1, 1]. Щоб це досягти, для кожної трійки матриць значень густини ρ, вільного члена S та тиску p ми виконали наступні дії:

$$\rho_{scaled} = \frac{\rho}{max\ (\rho_{max}, 1)} \tag{4.1}$$

$$S_{scaled} = \frac{S}{max\ (S_{max}, 1)} \tag{4.2}$$

$$p_{scaled} = \frac{p}{max\ (\rho_{max}, 1) * max\ (S_{max}, 1)} \tag{4.3}$$

де ρ_{max} і S_{max} – максимальні абсолютні значення густини та вільного члена рівняння відповідно. Таким чином ми забезпечили обмеження значень густини ρ і вільного члена S в діапазоні [-1, 1]. Але нові значення тиску p_{scaled} можуть бути поза цим діапазоном. Тому після генерування всього датасету, значення цільової змінної були нормалізовані до нормального розподілу з середнім значення 0 і стандартним відхиленням 1. Після нормалізації ці значення було відмасштабовано в діапазоні [-1, 1]. Тому, щоб перевести вихідні значення тренованої нейронної мережі у значення поля тиску необхідно зробити зворотнє масштабування і перехід від нормального розподілу до оригінального розподілу значень цільової змінної. Всі операції по нормалізації і масштабуванню були здійснені за допомогою бібліотеки scikit-learn [211].

Отриманий нами тренувальний датасет склав 60000 об’єктів. Тестовий датасет було сформовано аналогічним чином, який склав 10000 об’єктів.

Для реалізації та тренування нейронної мережі був використаний фреймворк для машинного навчання TensorFlow [208], версії 2.4.1. Для оптимізації параметрів нейронної мережі був використаний оптимізатор Adam [209], з відповідними параметрами навчання:

- $learning\_rate = 0{,}0005$
- $beta\_1 = 0{,}95$
- $beta\_2 = 0{,}99$
- $epsilon = 1e-7$

де $learning\_rate$ - параметр в алгоритмі оптимізації, який визначає розмір кроку оновлення коефіцієнтів нейронної мережі на кожній ітерації навчання; $beta\_1$ - коефіцієнт забування для градієнту; $beta\_2$ - коефіцієнт забування для другого

моменту градієнту; $epsilon$ – мала константа, що вводиться для забезпечення стабільності оптимізації. Ми обрали середньоабсолютну помилку як функцію втрат для даної моделі. Навчання здійснювалось на графічному процесорі MSI GeForce GTX 1660 Super Ventus OC 6GB GDDR6 протягом 300 епох і з розміром пакету 32 об’єкти.

Значення середньої абсолютної помилки тренованої моделі на тестовому наборі даних склало **0.000784.**

Таким чином, розроблена згорткова нейронна мережа є практичною реалізацією другого етапу запропонованого дворівневого методу.

4.2 Дворівневий метод моделювання руху рідини

Таким чином, для реалізації дворівневого підходу до моделювання руху рідини в закритих просторах використовуються дві сучасні технології: модифікована решітчаста модель Больцмана і згорткова нейронна мережа. Розроблений дворівневий метод реалізує схему предиктора та коректора, і кожна ітерація моделювання включає крок предиктора та крок коректора

Решітчаста модель Больцмана використовується для моделювання динаміки руху рідини у робочій області шляхом перетворення полів швидкостей з попереднього часового шару на полі швидкостей наступного шару, тобто реалізовує крок предиктора. Це досягається шляхом представлення рідини як елементарних об'ємів, які зіштовхуються вузлах дискретизації та розподіляються згідно з D2Q9 схемою з заданими ймовірностями. Оскільки решітчаста модель Больцмана призначена для роботи зі стисливими рідинами, вводиться проміжна швидкість для корекції. Ця корекція дозволяє точно визначити поле швидкостей рідини в робочій області, за умови, що рухається зі швидкістю, меншою за число Маха для даної рідини.

На наступному часовому шарі використовується згорткова нейронна мережа для вирішення рівняння Пуассона. Ця мережа попередньо навчена на тренувальному наборі даних, створеному для необхідних задач. Цей підхід

дозволяє отримати розподіл тиску на поточному часовому шарі з високою точністю. Таким чином розроблена нейронна мережа реалізовує крок коректора. За допомогою цього розподілу тиску визначається скориговане поле швидкостей, яке може бути використане як початкові дані для решітчастої моделі Больцмана при переході до визначення поля швидкостей на наступному часовому шарі.

Отже, загальний опис алгоритму даного методу наступний:

Вхідні дані: дискретизована обчислювальна область $\Omega = \{(x,y)|1 \le x \le N, 1 \le y \le N\}$, розміру $N \times N$, початкова густина ρ_0, параметр релаксації τ, швидкість звуку c_s, матриця кодування простору geo розміру $N \times N$, де $geo_{xy} = 1$ для твердого тіла, в іншому випадку $geo_{xy} = 0$, множина точок Ω_{out}, де реалізована гранична умова витоку, множина точок Ω_p, де реалізована гранична умова для тиску, множина точок Ω_{in}, де реалізована гранична умова притоку, значення початкової швидкості $\boldsymbol{u_{init}}(x,y)$, де $(x,y) \in \Omega$, значення граничної швидкості $\boldsymbol{u_{bc}}(x,y)$, де $(x,y) \in \Omega_{in}$, , значення граничних умов тиску $p_{bc}(x,y)$, де $(x,y) \in \Omega_p$

Крок 1. Розрахунок матриці відстаней до найближчої комірки твердого тіла $dist$ за допомоглю формули:

$$dist_{xy} = \frac{min_{(x_k,y_k)\in\Omega}\sqrt{(x-x_k)^2+(y-y_k)^2}}{N\sqrt{2}}, (x_k,y_k) \neq (x,y) \qquad (4.4)$$

Крок 2. Ініціалізація початкової макроскопічної швидкості $\boldsymbol{u}$ за допомогою початкової умови $\boldsymbol{u_{init}}(x,y)$ у всіх комірках обчислювальної області

Крок 3. Розрахунок значень модифікованої рівноважної функції розподілу $F_{eq}^{i}(x,y)$, де $(x,y) \in \Omega$ (2.90)

Крок 4. Застосування граничної умови витоку рідини для всіх комірок Ω_{out} (розділ 2.1.12)

Крок 5. Розрахунок макроскопічних величин - густини $\rho(x,y)$ і швидкості $\boldsymbol{u}(x,y)$, за допомогою формул (2.41) і (2.42) для кожної комірки Ω.

Крок 6. Застосування граничної умови притоку рідини для густини $\rho(x,y)$ і швидкості $\boldsymbol{u}(x,y)$ для всіх комірок $(x,y) \in \Omega_{in}$ (розділ 2.1.12)

Крок 7. Розрахунок тиску p за допомогою згорткової нейронної мережі для кожної комірки Ω (розділ 4.1)

Крок 8. Застосування граничних умов до отриманого поля тиску $p_{bc}(x,y)$, де $(x,y) \in \Omega_p$.

Крок 9. Корекція поля швидкості $\boldsymbol{u}(x,y)$, за допомогою формули (2.97)

Крок 10. Застосування граничної умови притоку рідини для значень функції розподілу $F^i(x,y)$, де $(x,y) \in \Omega_{in}$

Крок 11. Розрахунок модифікованої рівноважної функції розподілу $F_{eq}^i(x,y)$, де $(x,y) \in \Omega$ (2.90)

Крок 12. Оновлення функції розподілу за допомогою оператора зіткнення BGK та формули (2.25):

$$F_i\big((x,y),t+\Delta t\big) = F_i\big((x,y),t\big) - \frac{1}{\tau}\Big(F_i(x,y) - F_i^{eq}(x,y)\Big) \tag{4.5}$$

Крок 13. Застосування граничної умови твердої стіни для всіх (x,y), де $geo_{xy} = 1$ (розділ 2.1.12)

Крок 14. Крок поширення (2.40):

$$F_i\Big(\big(x + c_{ix}\Delta t, y + c_{iy}\Delta t\big), t+\Delta t\Big) = F_i\big((x,y),t+\Delta t\big) \tag{4.6}$$

Крок 15. Перейти до кроку 4, якщо моделювання не завершено

4.3 Паралельний метод моделювання руху рідини на основі LBM

Також ми розробили метод розпаралелювання обчислень для розробленого методу, оскільки в даному випадку воно дає нам наступні переваги:

1. Паралельне виконання значно скорочує час моделювання, що призводить до підвищення ефективності обчислень і швидших результатів.
2. Масштабованість розпаралелювання дозволяє ефективно масштабувати LBM зі збільшенням розміру обчислювальної області. Розподіляючи обчислювальне навантаження між декількома процесорами,

паралельні реалізації LBM можуть обробляти більші сітки, складнішу геометрію та моделювання з вищою роздільною здатністю. Ця масштабованість є важливою для моделювання реальних програм із складною поведінкою потоку або великими обчислювальними областями.

3. Використання високопродуктивних обчислювальних систем (англ. High Performance Computing, HPC). Паралельні реалізації LBM можуть ефективно використовувати можливості високопродуктивних обчислювальних систем, таких як кластери, суперкомп'ютери або багатоядерні процесори. Ці системи забезпечують велику кількість процесорів або обчислювальних вузлів, які можна використовувати одночасно для прискорення моделювання LBM, що дозволяє аналізувати більш широкі та детальніші явища потоку рідини.

4. Кілька симуляцій з різними конфігураціями параметрів або вхідними умовами можуть виконуватися паралельно, що дозволяє ефективно досліджувати простір параметрів. Ця можливість особливо цінна для задач оптимізації, кількісної оцінки невизначеності або аналізу чутливості, де потрібно виконати численні симуляції.

5. Симуляції в реальному часі мають вирішальне значення в різних сферах, таких як віртуальне прототипування, віртуальне тестування або інтерактивне моделювання, де швидкі реакції потрібні для процесів прийняття рішень.

Розроблений нами метод має в своїй основі підхід під назвою domain decomposition [212].

Методи domain decomposition — це чисельні методи, які використовуються для розв'язування диференціальних рівнянь із частинними похідними або виконання моделювання у великих обчислювальних областях шляхом поділу їх на менші підобласті або частини. Основною метою domain decomposition є розподіл обчислювального навантаження між кількома процесорами або обчислювальними вузлами, що забезпечує паралельну обробку та ефективне використання ресурсів високопродуктивних обчислень.

У методах domain decomposition обчислювальна область розкладається на підобласті або підобласті, які не перекриваються, які можуть оброблятися незалежно. Кожен піддомен зазвичай відповідає окремій підмножині сітки або сітки, що використовується для дискретизації проблеми. Декомпозицію можна виконувати різними способами, наприклад прямокутними або криволінійними поділами, структурованими чи неструктурованими сітками або ієрархічними декомпозиціями.

Процес domain decomposition включає кілька ключових етапів:

1. Обчислювальна область поділяється на менші підобласті. Розбиття можна виконати за допомогою різних стратегій, таких як геометричне розбиття, розбиття графа або рекурсивне поділ. Мета полягає в тому, щоб рівномірно розподілити навантаження та мінімізувати накладні витрати на зв’язок між підобластями.
2. Сусідні підобласті повинні обмінюватися інформацією для забезпечення дотримання граничних умов або вирішення пов’язаних проблем. Зв’язок між підобластями може бути досягнутий за допомогою передачі повідомлень, коли дані обмінюються між процесорами за допомогою бібліотек зв’язку, таких як інтерфейс передачі повідомлень MPI [45]. Належні схеми й алгоритми зв’язку є важливими для забезпечення точних і послідовних рішень у підобластях.
3. Кожна підобласть розв’язує диференціальні рівняння в частинних похідних або виконує моделювання незалежно в межах власного обчислювального домену. Це може включати розв’язування диференціального рівняння або іншого обчислення за допомогою чисельних методів, таких як методи скінченних різниць, скінченних елементів або скінченних об’ємів. Локальний розв’язок у кожній підобласті зазвичай отримується ітераційно, доки не буде досягнуто збіжності.
4. Рішення з сусідніх підобластей потрібно з’єднати або об’єднати на інтерфейсах між підобластями, щоб забезпечити глобальну узгодженість і безперервність. Це включає в себе обмін даними або значеннями рішення на

кордонах суміжних підобластей. Для обробки зв'язку інтерфейсу використовуються різні методи з'єднання, такі як методи перекривання або фантомних комірок.

5. Методи domain decomposition часто вимагають ітераційного процесу для досягнення глобального рішення. Рішення підобластей оновлюються ітераційно, а зв'язок між підобластями коригується, поки не буде отримано узгоджене глобальне рішення. Ітерації тривають, доки не буде досягнуто бажаної точності або критеріїв збіжності.

На основі описаного підходу ми розробили окремий метод для розпаралелювання розробленого дворівневого методу моделювання руху рідин на квадратній дискретизованій обчислювальній області, розміру $N \times N$, зображеній на рис. 4.5., для довільного числа процесорів p.

1,1	2,1	3,1	...	N-1,1	N,1
1,2	2,2	3,2	...	N-1,2	N,2
1,3	2,3	3,3	...	N-1,3	N,3
...	...	...	...	...	...
1,N-1	2,N-1	N-1,3	...	N-1,N-1	N,N-1
1,N	2,N	N,3	...	N-1,N	N,N

Рисунок 4.5 – Дискретизована обчислювальна область розміру $N \times N$

Перший крок – це розбиття обчислювальної області на частини, кількість яких дорівнює кількості процесорів. Поділ відбувається на рівні по ширині прямокутники – ширина i-ої підобласті $w_i = N$. Висота підобластей обчислюється

наступним чином: якщо N ділиться на p без остачі, тоді висота $h_i = N/p$, інакше $h_i = \lfloor N/p \rfloor + s_i$, де $i = 1, \dots, p$, та

$$s_i = \begin{cases} 1, & i \leq N \bmod p \\ 0, & i > N \bmod p \end{cases} \quad (4.7)$$

Таким чином ми отримали обчислювальні області, які максимально наближено рівні за кількістю комірок в кожній, для довільної квадратної області і довільної кількості процесорів. Приклад розбиття області розміром 16×16 зображений на рис. 4.6.

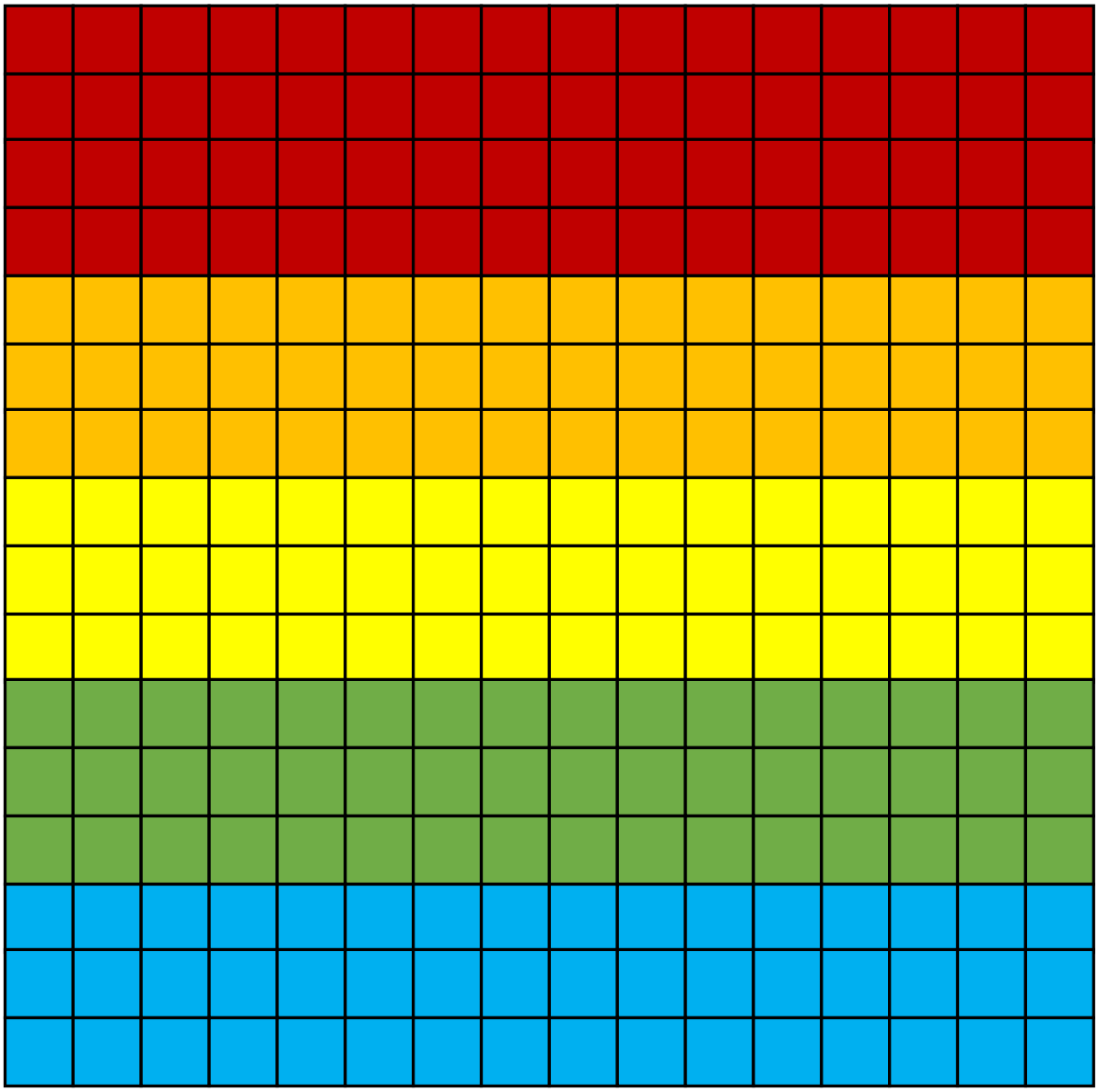

Рисунок 4.6 – Розбиття області розміром 16×16 на підобласті при $p = 5$

Далі, для кожної підобласті виконується алгоритм дворівневого методу, описаний в кінці розділу 4.2, з наступними модифікаціями. Крок 10 неможливо розпаралелити для однієї ітерації застосування нейронної мережі. Тому кожен

процесор, що моделює обчислювальну область, обчислює свою частину вільного члена S, та пересилає в процесор, що розраховує значення поля тиску за допомогою нейронної мережі, значення S, а також значення густини в комірках своєї області ρ. Після розрахунку значень тиску, за допомогою розробленої нейронної мережі, ці значення пересилаються у інші процесори, відповідно до обчислювальної області.

Іншим важливим аспектом для правильної побудови паралельного алгоритму на основі методу domain decomposition є те, що при розрахунку значень вільного члена S_{ij} використовуються значення макроскопічної швидкості $\boldsymbol{u}$ в сусідніх комірках. І тому, перед розрахунком вільного члена, кожному процесору необхідно отримати значення поля швидкості із граничних комірок сусідніх областей, що зображено на рис. 4.7.

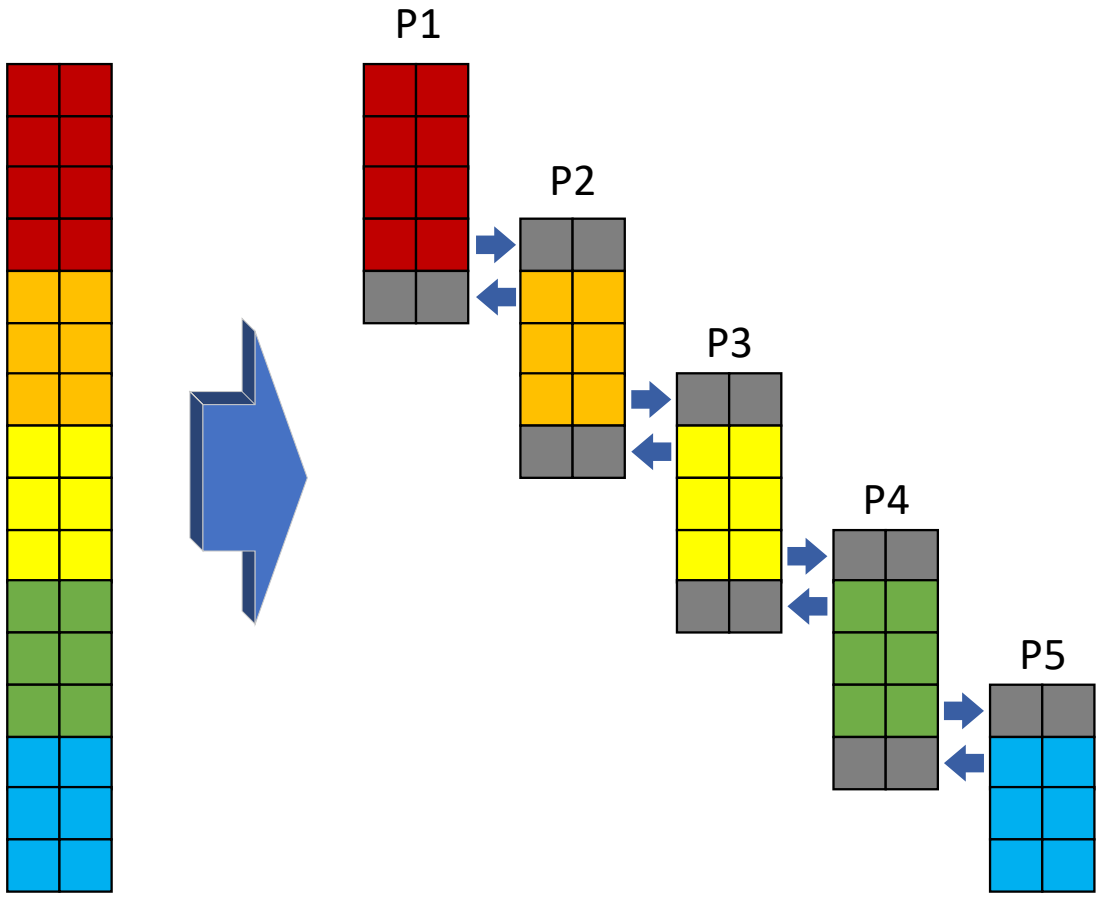

Рисунок 4.7 – Схема обміну даними між сусідніми підобластями та відповідними процесорами

На основі принципу, що зображений на рис 4.7, також відбувається обмін значеннями функцій розподілу між областями перед кроком 13, тобто кроком поширення, дворівневого методу, оскільки для кожної комірки з координатами

(x, y), необхідно значення із сусідних комірках, як і у випадку з макроскопічною швидкістю. Так, згідно з моделлю *D2Q9*, значення функцій розподілу $F_{4,7,8}$ нижніх комірок області i будуть переслані в область $i + 1$, а значення функцій розподілу $F_{2,5,6}$ верхніх комірок області i будуть переслані в область $i - 1$.

Таким чином загальний алгоритм для кожного процесора і обчислювальної області i буде наступний:

1. Виконати кроки 1-3 алгоритму дворівневого методу
2. Виконати кроки 4-6 алгоритму дворівневого методу
3. Переслати значення макроскопічної швидкості у сусідні області $i - 1$ та $i + 1$
4. Розрахувати значення вільного члена S
5. Якщо процесор виконує обчислення поля тиску за допомогою нейронної мережі, то отримати значення вільного члену та густини від інших процесорів, застосувати нейронну мережу для обчислення поля тиску, переслати відповідні значення у відповідні процесори
6. Якщо процесор не виконує обчислення поля тиску, то відправити значення вільного члену та густини в процесор, що розраховує значення поля тиску, та отримати значення поля тиску відповідно до обчислювальної області
7. Виконати кроки 9-13 алгоритму дворівневого методу
8. Переслати значення функції розподілу у сусідні області $i - 1$ та $i + 1$
9. Виконати крок 14 алгоритму дворівневого методу
10. Перейти до кроку 2 цього алгоритму ,якщо моделювання не завершено

Розроблений метод теоретично дозволяє масштабувати розроблений дворівневий метод моделювання руху рідини для обчислювальних систем з довільною кількістю процесорів. Представлений метод розбиття обчислювальної області дозволяє звести кількість пересилок до мінімуму, оскільки кожна обчислювальна область має 1 чи 2 сусідні області.

4.4 Апаратна складова

Сучасні програмні засоби дозволяють використовувати звичайні процесори для роботи нейронних мереж. Але швидкість обчислень в такому випадку є меншою, ніж при використанні спеціалізованого апаратного забезпечення. Оскільки ключовою складовою нашого методу є згорткова нейронна мережа, підбір та розробка спеціального прискорювача для збільшення швидкості роботи нейронної мережі є ключовою.

Апаратні прискорювачі для згорткових нейронних мереж — це спеціалізовані обчислювальні пристрої, призначені для прискорення обчислень, пов'язаних з операціями згорткових нейронних мереж. Ці прискорювачі спеціально оптимізовані для паралелізму та шаблонів потоку даних, присутніх у згорткових нейронних мережах, що забезпечує швидшу та ефективнішу обробку задач таких як розпізнавання зображень і відео, виявлення об'єктів і обробка природної мови. Ось короткий огляд деяких популярних апаратних прискорювачів для згорткових нейронних мереж:

- Графічні процесори (англ. graphics processing unit, GPU) - спочатку GPU були розроблені для візуалізації комп'ютерної графіки, але знайшли широке застосування в глибокому навчанні завдяки їхнім можливостям паралельної обробки. Сучасні графічні процесори мають тисячі ядер, які можуть ефективно виконувати матричні операції, що робить їх добре придатними для обчислень нейронних мереж. Такі фреймворки, як CUDA [146] та cuDNN [213], дозволяють ефективно використовувати GPU для завдань глибокого навчання.
- Тензорні блоки обробки (англ. tensor processing unit, TPU) — це спеціально розроблені прискорювачі, розроблені Google [214] спеціально для задач глибокого навчання. Вони чудово справляються з операціями множення матриць, які є фундаментальними для обчислень згорткових нейронних мереж. TPU відомі своєю високою пропускною здатністю та

енергоефективністю, що робить їх популярним вибором для масштабних завдань машинного навчання.

- Програмовані користувачем вентильні матриці (англ. Field-Programmable Gate Array, FPGA) — це реконфігуровані апаратні пристрої, які можна налаштувати для виконання конкретних обчислень згорткової нейронної мережі. Вони забезпечують гнучкість з точки зору дизайну та можуть досягти високої продуктивності за рахунок паралелізму. FPGA зазвичай використовуються для прискорення роботи згорткових нейронних мереж у випадках, де енергоефективність і низька затримка обробки є критичними, наприклад, периферійні обчислення та програми реального часу.
- Інтегральні схеми для специфічного застосування (англ. Application-specific integrated circuit, ASIC) — це спеціально розроблені мікросхеми, створені спеціально для прискорення робочих навантажень глибокого навчання. На відміну від FPGA, ASIC є пристроями з фіксованою функцією та пропонують вищу продуктивність і енергоефективність за рахунок оптимізації апаратної архітектури для конкретних операцій згорткової нейронної мережі. ASIC можуть розроблятися спеціалізованими апаратними компаніями або організаціями зі значними ресурсами, такими як великі технологічні компанії.
- Нейронний процесор (англ. Neural Processing Unit, NPU) — це спеціалізовані апаратні блоки, призначені для ефективного виконання обчислень нейронної мережі. Зазвичай вони інтегровані в системи на кристалі для мобільних і вбудованих пристроїв. NPU часто мають спеціальну схему для операцій нейронних мереж і забезпечують низькоенергетичні та високопродуктивні рішення для програм глибокого навчання на пристрої.
- Процесори цифрової обробки сигналів (англ. digital signal processor, DSP) — це програмовані процесори, оптимізовані для обробки завдань цифрової обробки сигналів. Вони широко використовуються в різних областях, включаючи глибоке навчання. DSP можна використовувати для

прискорення обчислень нейронних мереж шляхом використання їх можливостей паралельної обробки та ефективної обробки чисел.

Переваги та недоліки використання кожного типу апаратного забезпечення для прискорення роботи згорткових мереж:

1. GPU

Переваги:

- Високі можливості паралельної обробки з тисячами ядер.
- Широка доступність і підтримується фреймворками глибокого навчання.
- Хороша продуктивність для обчислень згорткових нейронних мереж, особливо для великомасштабних моделей.

Недоліки:

- Високе енергоспоживання порівняно зі спеціалізованими прискорювачами.
- Обмежена пропускна здатність пам’яті для певних операцій.
- Не розроблено спеціально для глибокого навчання, що в деяких випадках призводить до неоптимальної ефективності.

2. TPU

Переваги:

- Спеціально створений для навантажень глибокого навчання, що забезпечує високу продуктивність.
- Оптимізовано для операцій множення матриць.
- Відмінна енергоефективність і висока пропускна здатність.

Недоліки:

- Обмежена доступність, переважно використовується в службах Google Cloud.
- Менша гнучкість порівняно з графічним процесором або FPGA через фіксовану архітектуру.
- Обмежена підтримка обчислювальних завдань загального призначення

3. FPGA

Переваги:

- Апаратне забезпечення з можливістю налаштування та реконфігурації.
- Може досягти високої продуктивності та низької затримки для певних операцій нейронних мереж.
- Енергоефективне і добре підходить для сценаріїв периферійних обчислень.

Недоліки:

- Складність дизайну та програмування порівняно з готовими прискорювачами.
- Обмежений обсяг пам'яті та пропускна здатність.
- Вища вартість і довший час розробки порівняно з GPU або TPU.

4. ASIC

Переваги:

- Спеціально розроблене апаратне забезпечення для глибокого навчання, що забезпечує високу продуктивність.
- Оптимізовано для конкретних операцій нейронних мереж, що забезпечує ефективну обробку.
- Менше енергоспоживання в порівнянні з іншими прискорювачами.

Недоліки:

- Дорогий у розробці та виготовленні.
- Відсутність гнучкості через конструкцію з фіксованими функціями.
- Довший цикл розробки та обмежена доступність порівняно з іншими прискорювачами.

5. NPU

Переваги:

- Інтегровано в мобільні та вбудовані пристрої, що дозволяє робити висновки на пристрої.
- Низьке енергоспоживання та висока продуктивність для обчислень нейронних мереж.

- Ефективний для периферійних обчислень і програм реального часу.

Недоліки:

- Обмежена доступність на певних пристроях або платформах.
- Менша гнучкість порівняно з графічним процесором або FPGA.
- Може мати обмежену підтримку для обчислювальних завдань загального призначення.

6. DSP

Переваги:

- Широко доступний і широко використовується для завдань обробки сигналів.
- Хороші можливості паралельної обробки.
- Нижче енергоспоживання порівняно з графічним процесором.

Недоліки:

- Зазвичай не настільки ефективні, як спеціалізовані прискорювачі глибокого навчання.
- Обмежений обсяг пам'яті та пропускна здатність.
- Обмежена підтримка певних фреймворків глибокого навчання та бібліотек.

Приклади апаратних прискорювачів кожного згаданого типу зображені на рис. 4.8.

Враховуючи всі переваги та недоліки кожного типу описаного апаратного забезпечення, ми вирішили реалізувати нейронну мережу для двох типів прискорювачів: GPU та NPU. Оскільки, для тренування нейронних мереж ми використали фреймворк Tensorflow та програмне забезпечення CUDA та cuDNN, то розроблена модель вже оптимізована для графічних процесорів компанії nVidia. Тому для експериментів ми використали той же графічний процесор, на якому виконувалось навчання нейронної мережі.

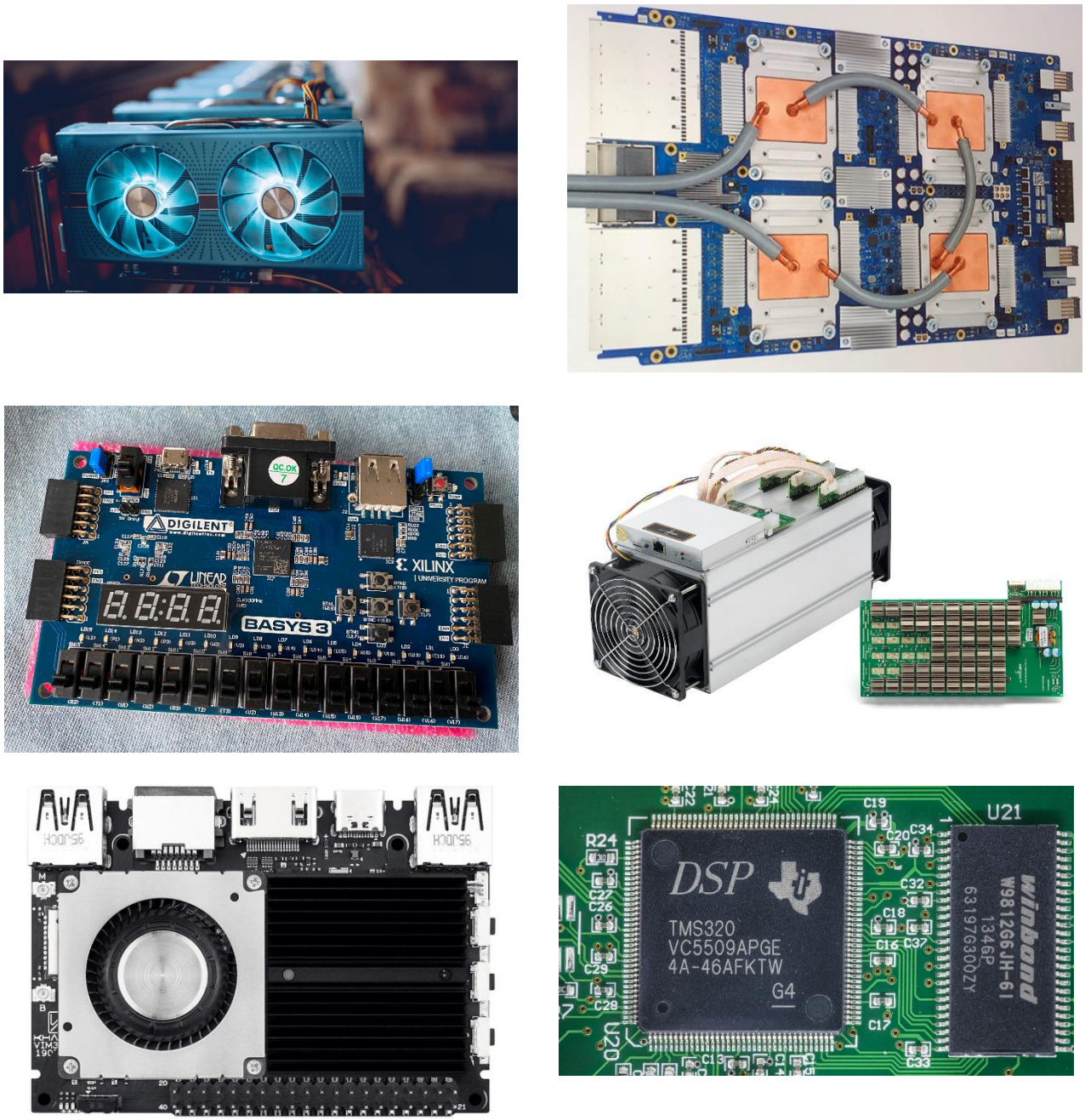

Рисунок 4.8 – Різні типи апаратного забезпечення, що може бути використано для прискорення роботи нейронних мереж. Зліва направо, зверху вниз: GPU, TPU, FPGA, ASIC, NPU, DSP

Для реалізації на NPU ми вибрали плату Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit [215] компанії AMD [216] (рис. 4.9). Даний вибір обгрунтований кількома факторами: наявність спеціалізованого програмного забезпечення від виробника, що дозволяє швидко оптимізувати розроблені нейронні мережі, за допомогою популярних фреймворків машинного навчання, для архітектури

цільового апаратного забезпечення, високі показники продуктивності та обчислювального швидкості, помірна ціна. Короткі технічні характеристики апаратного забезпечення описані в таблиці 4.2. Структурна схема співпроцесора NPU, що реалізований на обраній платі, зображена на рис 4.10.

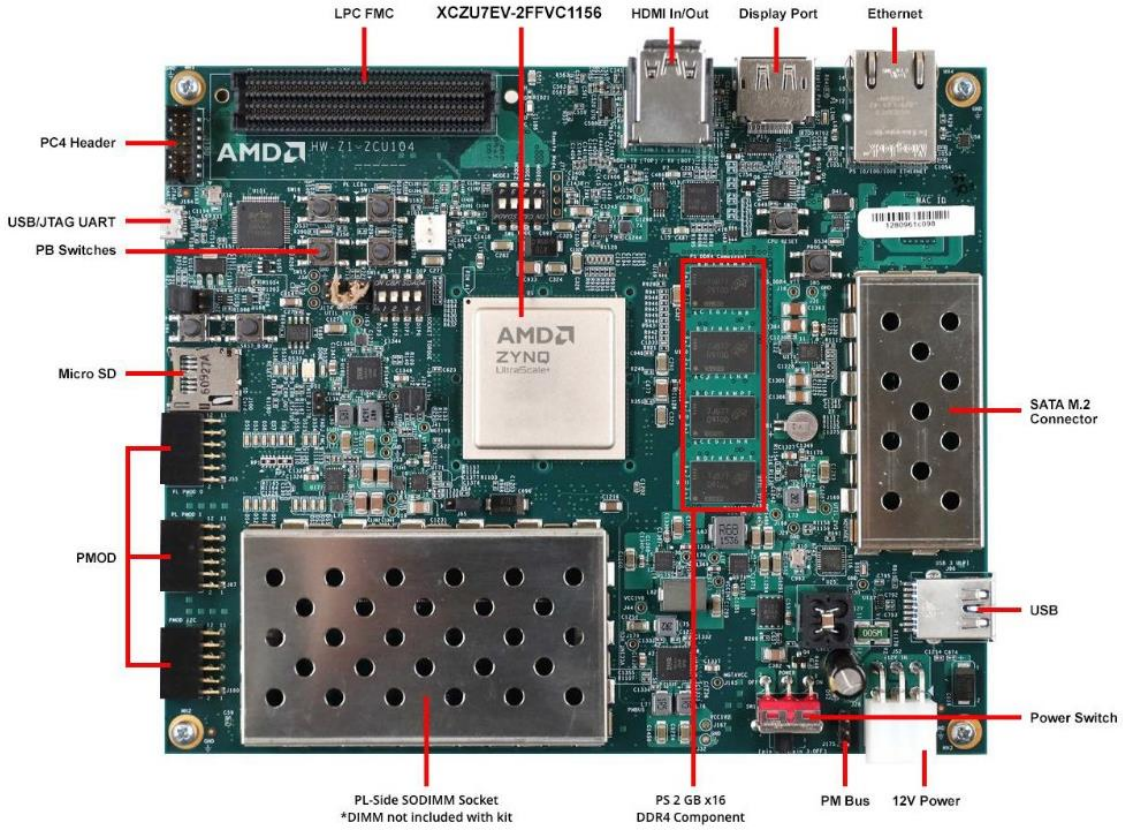

Рисунок 4.9 – плата Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit для програмування NPU

Таблиця 4.2 – Характеристики Zynq UltraScale+ MPSoC ZCU104

Процесори	Cortex-A53 Arm v8 architecture-based 64-bit, Cortex-R5 Arm v7 architecture-based 32-bit
Накопичувальні пристрої	PS DDR4 64-bit, Quad-SPI flash, Micro SD card slot
Порти вводу-виводу	USB-UARTs with FT4232H JTAG/3xUART Bridge, RJ-45 Ethernet connector, SATA (M.2) for SSD access

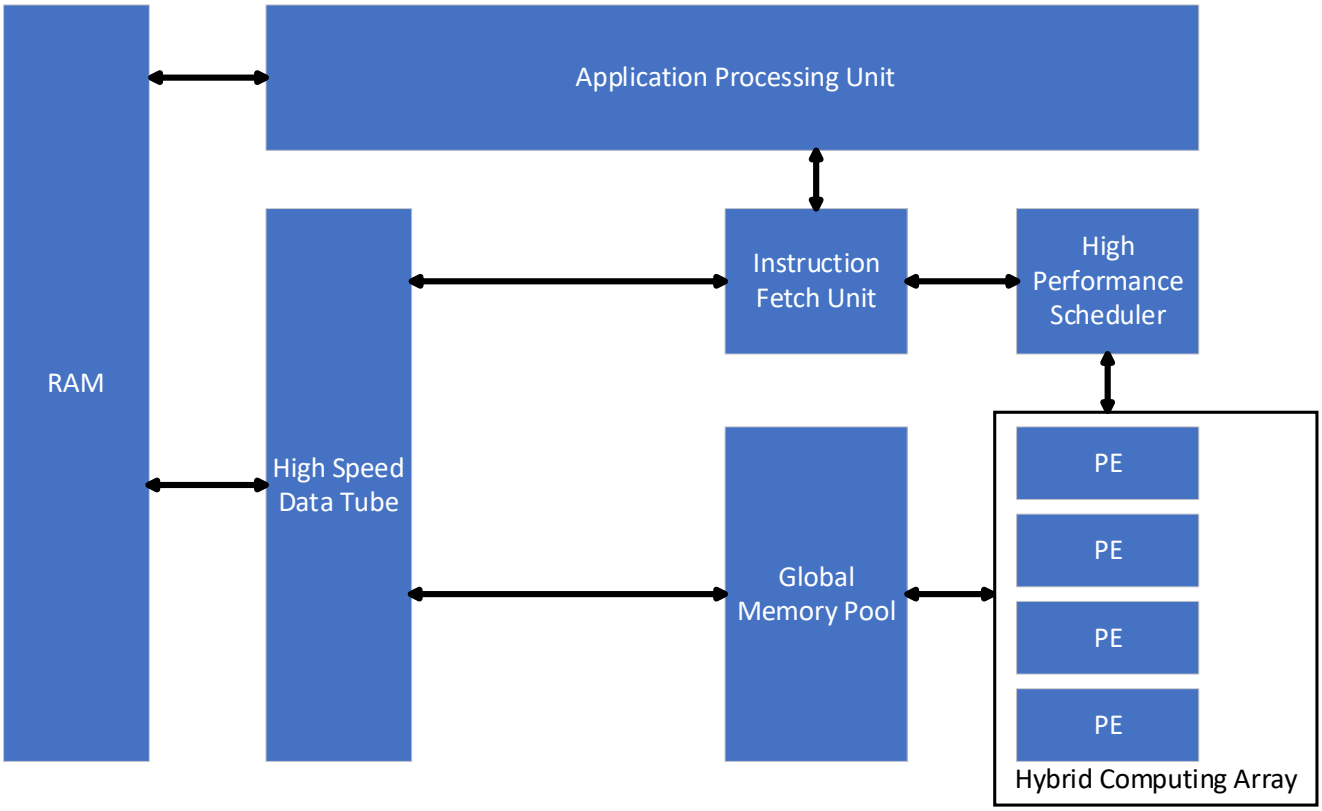

Рисунок 4.10 – Структурна схема співпроцесора NPU

Співпроцесор складається з наступних складових: пул глобальної пам'ятті (Global Memory Pool), планувальника обчислень (High Performance Scheduler), гібридного масиву обчислювачів (Hybrid Computing Array), пристрою вибору інструкцій (Instruction Fetch Unit), блоку керування (Application Processing Unit) та каналу передачі даних (High Speed Data Tube). Даний співпроцесор може бути настроєний під нейронну мережу довільної структури.

Першим кроком є прунінг тренованої за допомогою фреймворку Tensorflow моделі, під цільову архітектуру за допомогою програмного забезпечення Vitis AI [218] від виробника апаратного забезпечення. Прунінг нейронної мережі відноситься до процесу вибіркового видалення з'єднань або нейронів із мережі, щоб зменшити її розмір і обчислювальну складність, зберігаючи або покращуючи її продуктивність. Прунінг — це техніка, яка використовується для стиснення нейронної мережі та оптимізації моделі. У контексті нейронних мереж прунінг спрямований на усунення зайвих або непотрібних параметрів, зв'язків або нейронів, що може призвести до більш ефективних і легких моделей. Програмний код, що використовує засоби для прунінгу зображений на рис. 4.11.

Другим кроком стало конвертація стисненої мережі у формат, який прийнятний для квантизації моделі. Код, що виконує цей процес зображений на рис. 4.12.

```
model = tf.keras.models.load_model('./poisson_cnn.h5')
from tensorflow_model_optimization.quantization.keras import vitis_inspect
inspector = vitis_inspect.VitisInspector(target=" XCZU7EV-2FFVC1156")
inspector.inspect_model(model,
                        dump_model=True,
                        dump_model_file="./pruned_model.h5")
```

Рисунок 4.11 – Фрагмент коду прунінгу тренованої моделі

```
from tensorflow import keras
poisson_model = keras.models.load_model("./pruned_model.h5")
poisson_model.save("poisson_model")
```

Рисунок 4.12 – Фрагмент коду конвертації стисненої мережі

Третім кроком є квантизація моделі. Квантизація нейронної мережі — це процес, який передбачає зменшення точності або розрядності числових значень, які використовуються для представлення параметрів мережі та активацій. Він має на меті представити мережу за допомогою типів даних нижчої точності, таких як цілі чи числа з фіксованою комою, замість типів даних вищої точності, таких як числа з плаваючою комою. Ця техніка використовується для зменшення обсягу пам’яті, обчислювальної складності та споживання енергії, зберігаючи при цьому прийнятну продуктивність.

У квантизації дійсні ваги, зміщення та активації нейронної мережі апроксимуються за допомогою зменшеної кількості бітів. Це зниження точності пов’язане з певними компромісами, але може запропонувати значні переваги, особливо для розгортання на пристроях з обмеженими ресурсами, мобільних платформах або вбудованих системах.

Ми квантизували нейронну мережу, яка використовує перехід від 32-бітових чисел з плаваючою комою до 8-бітових цілочисельних значень. Код, що виконує квантизацію зображений на рис. 4.13.

Наступним кроком є компіляція квантизованої моделі для цільової апаратної платформи. Вона відбувається за допомогою згаданого програмного забезпечення Vitis AI. Фрагмент коду для компіляції нейронної мережі зображений на рис. 4.14.

Після компіляції модель готова до запуску на цільовому апаратному забезпеченні.

```
from tensorflow import keras
poisson_model = keras.models.load_model("./poisson_model")
from tensorflow_model_optimization.quantization.keras import vitis_quantize
quantizer = vitis_quantize.VitisQuantizer(poisson_model, quantize_strategy='fs')
quantized_model = quantizer.quantize_model(calib_dataset=calib_dataset)
quantized_model.save('./quantized_model')
```

Рисунок 4.13 – Фрагмент коду квантизації нейронної мережі

```
$     vai_c_tensorflow \
>         --frozen_pb=./build/quantized_results/poisson_net/quantize_model.pb \
>         --arch /opt/vitis_ai/compiler/arch/DPUCZDX8G/ZCU104/arch.json \
>         --output_dir=./build/compile/poisson_net \
>         --net_name=poisson_net \
>         --options    "{'mode':'normal'}"
```

Рисунок 4.14 – Фрагмент коду компіляції нейронної мережі для цільового апаратного забезпечення

4.5 Програмна реалізація

На базі описаного модифікованого методу LBM, опису задання початкових та граничних умов, розробленої та натренованої нейронної мережі, було розроблене програмне забезпечення, яке дозволило дослідити розроблений метод моделювання руху рідин при різних вхідних параметрах.

Програмне забезпечення було реалізовано за допомогою мови програмування Python, бібліотеки програмування графічного користувацького інтерфейсу PyQt6 [218], бібліотеки математичних операцій над багатовимірними масивами NumPy [219], бібліотеки візуалізації даних matplotlib [220], бібліотеки роботи з растровою графікою PIL [221] та фреймворку для глибокого машинного навчання TensorFlow.

Розроблене програмне забезпечення містить наступні функції:

- встановлення фізичних параметрів рідини: початкову густину, параметр релаксації
- граничні умови: швидкість притоку рідини, напрям притоку і витоку (як одна із чотирьох сторін)

- опції корекції швидкості
- вибір апаратного прискорювача у випадку корекції швидкості за допомогою поля тиску, яке було отримане за допомогою нейронної мережі
- кількість ітерацій моделювання
- можливість завантаження файлу з геометрією обчислювальної області. Даний файл повинен бути картинкою формату .png або .jpg, розміру 96 × 96 пікселів, і мати тільки 2 кольори – чорний, що визначає простір заповнений рідиною, і білий – простір, що заповнений твердим тілом.
- Кнопки для запуску та зупинки моделювання, та для збереження результатів моделювання для подальшого аналізу.
- Візуалізація результатів успішного моделювання

Інтерфейс програми після її запуску зображений на рис. 4.15.

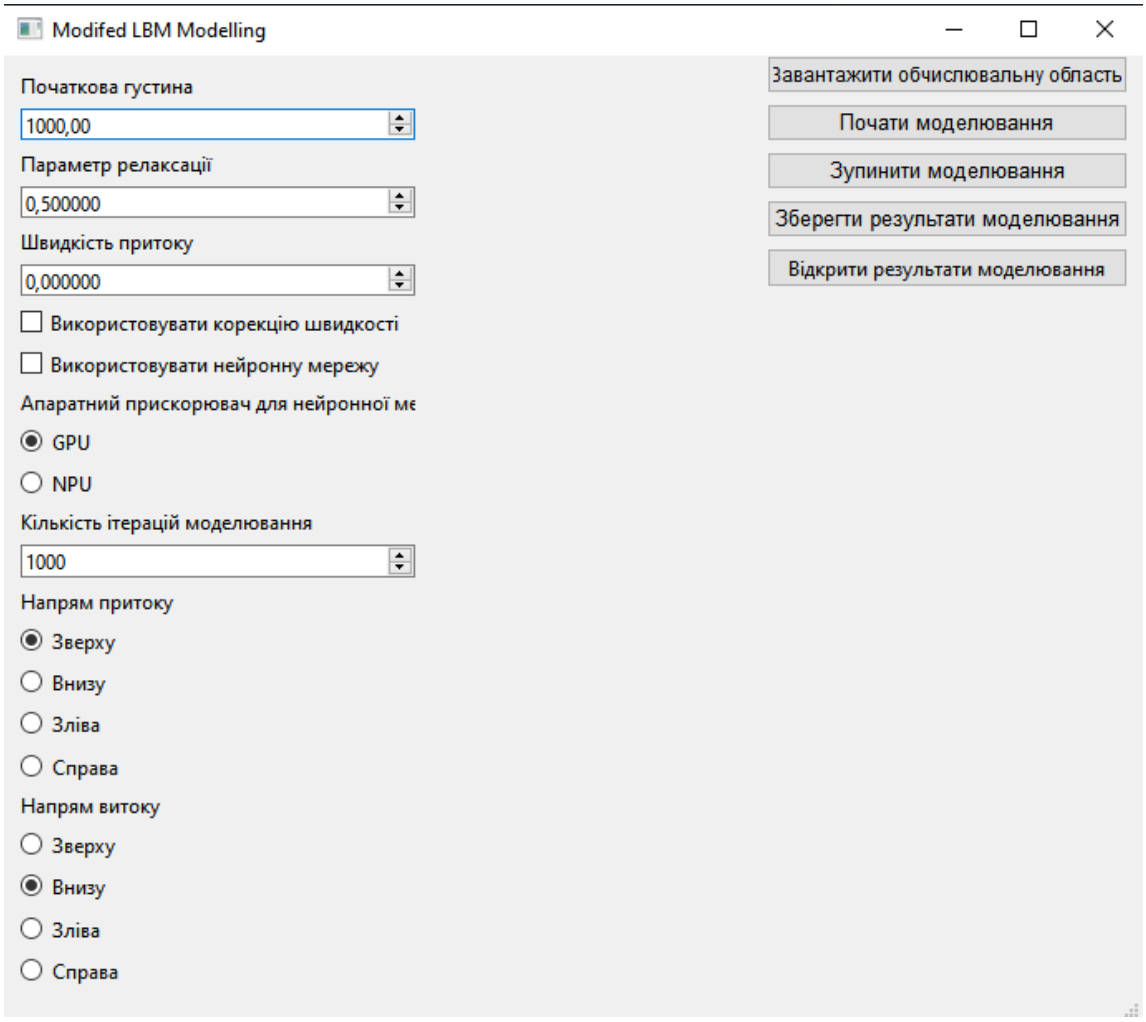

Рисунок 4.15 – Програма для моделювання руху рідин за допомогою модифікованого дворівневого методу на основі LBM

Після запуску програми, першим кроком є завантаження файлу з геометрією обчислювальної області, у вигляді чорно-білого растрового зображення. Приклад таких зображень на рис. 4.16. Стан вікна інтерфейсу програми після завантаження файлу на рис. 4.17.

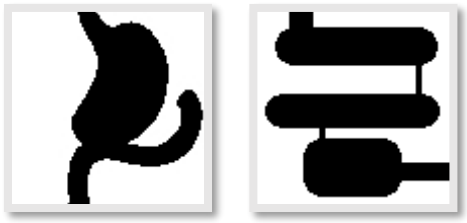

Рисунок 4.16 – Приклад зображень геометрій простору

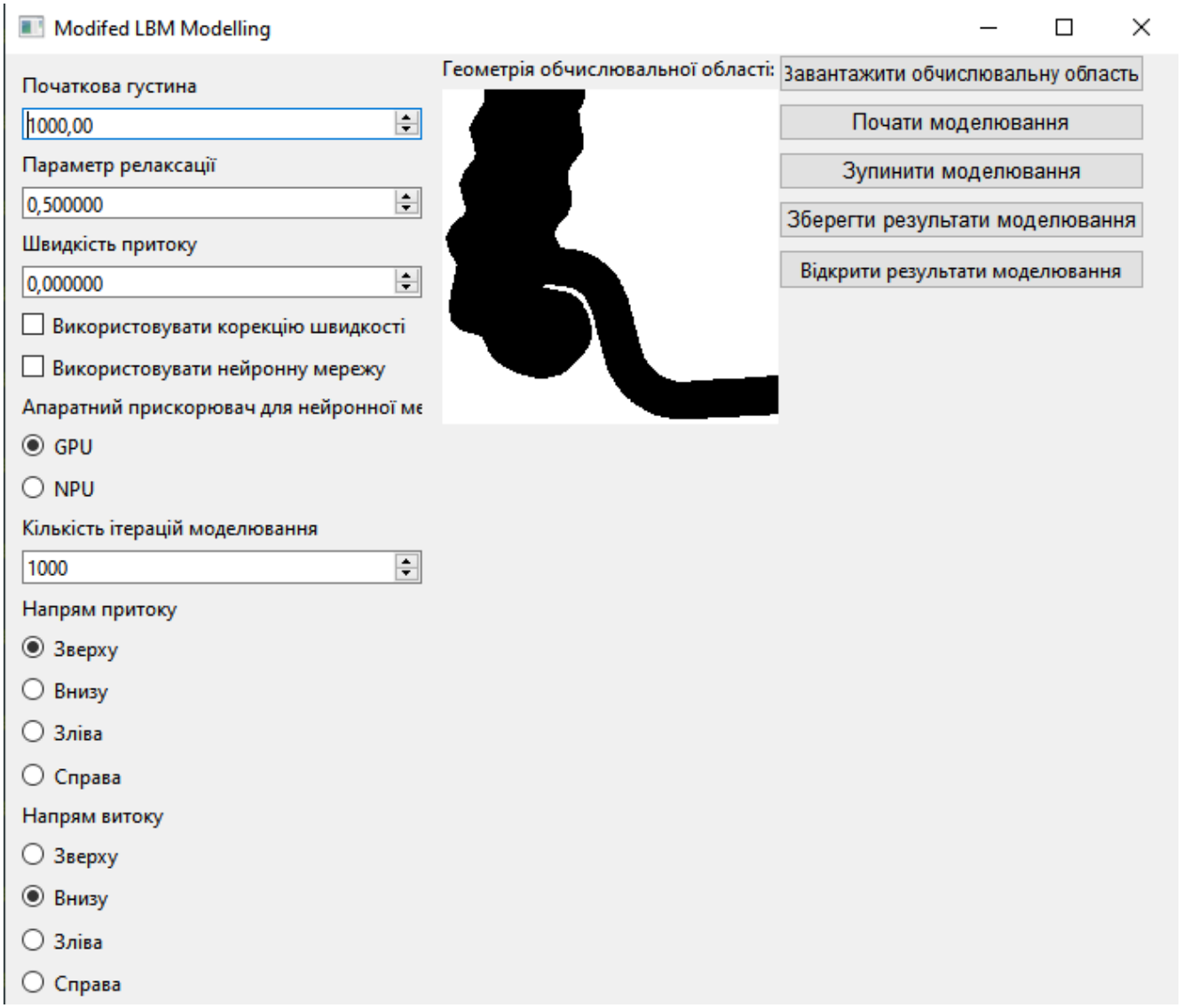

Рисунок 4.17 – Стан програми після завантаження обчислювальної області

Після встановлення значень параметрів моделювання (ліва частина вікна програми), можна почати моделювання, натиснувши кнопку «Почати

моделювання». Якщо необхідно перервати моделювання з будь яких причин – необхідно натиснути кнопку «Зупинити моделювання».

Під час моделювання з’явиться шкала прогресу, яка демонструє скільки відсотків від заданої кількості ітерацій було промодельовано (рис. 4.18).

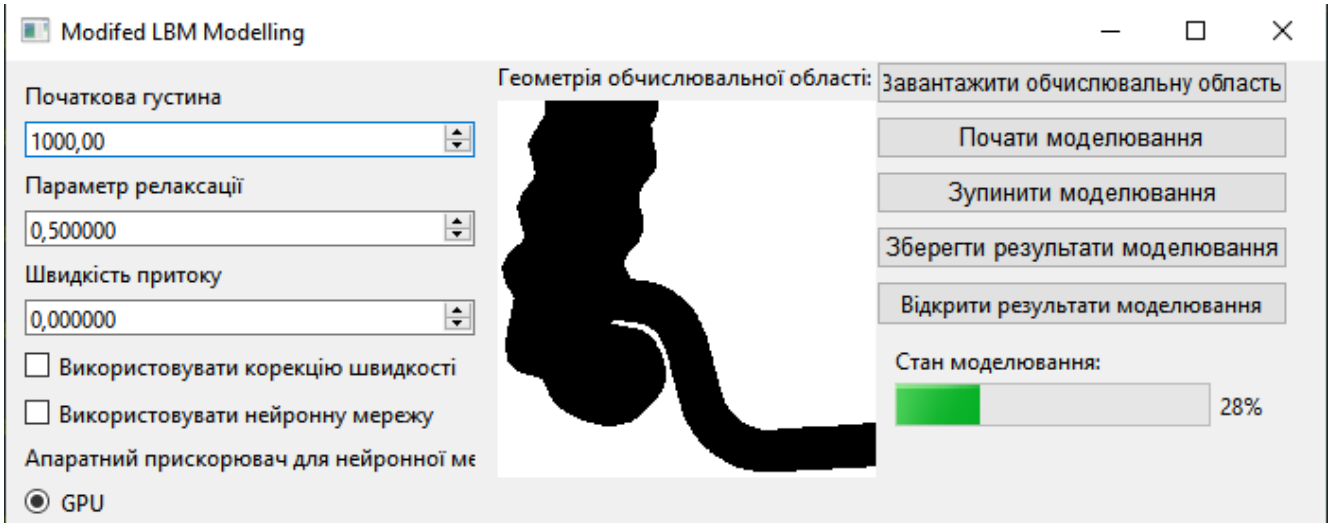

Рисунок 4.18 – Процес моделювання руху рідин

Після успішного моделювання всієї кількості заданих ітерацій, шкала прогресу заповниться до 100%. Після цього результати моделювання можна зберегти в окрему папку, для подальшої візуалізації необхідних фізичних величин, натиснувши кнопку «Зберегти результати моделювання». Результом збереженого моделювання є п’ять файлів:

- Дані про макроскопічну швидкість на всій області моделювання, формат .npy [221]
- Дані про густину, формат .npy
- Дані про тиску, формат .npy
- Дані про час затрачений на моделювання кожної ітерації, формат .csv [222]
- Дані про вхідні параметри моделювання, формат .json [223]

Після цього результати можна візуалізувати, натиснувши кнопку «Відкрити результати моделювання», та вибрати папку, де збережені дані моделювання. Після цього відкриється нове вікно, де можна ознайомитись з візуалізацією значень полей густини, швидкості та тиску для обраної ітерації моделювання (рис. 4.20).

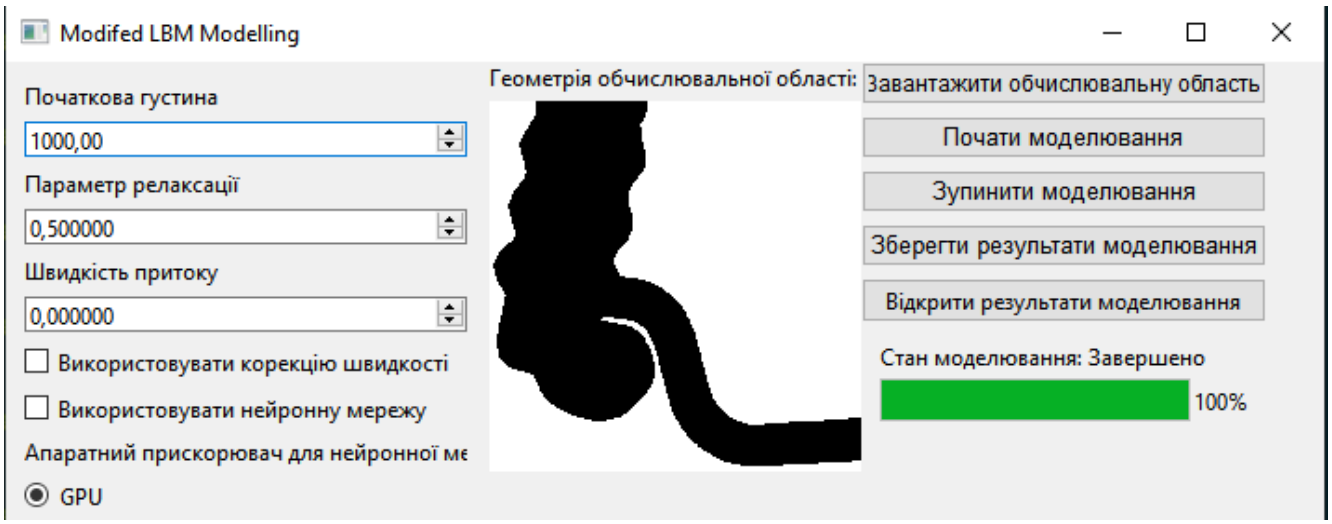

Рисунок 4.19 – Завершений процес моделювання

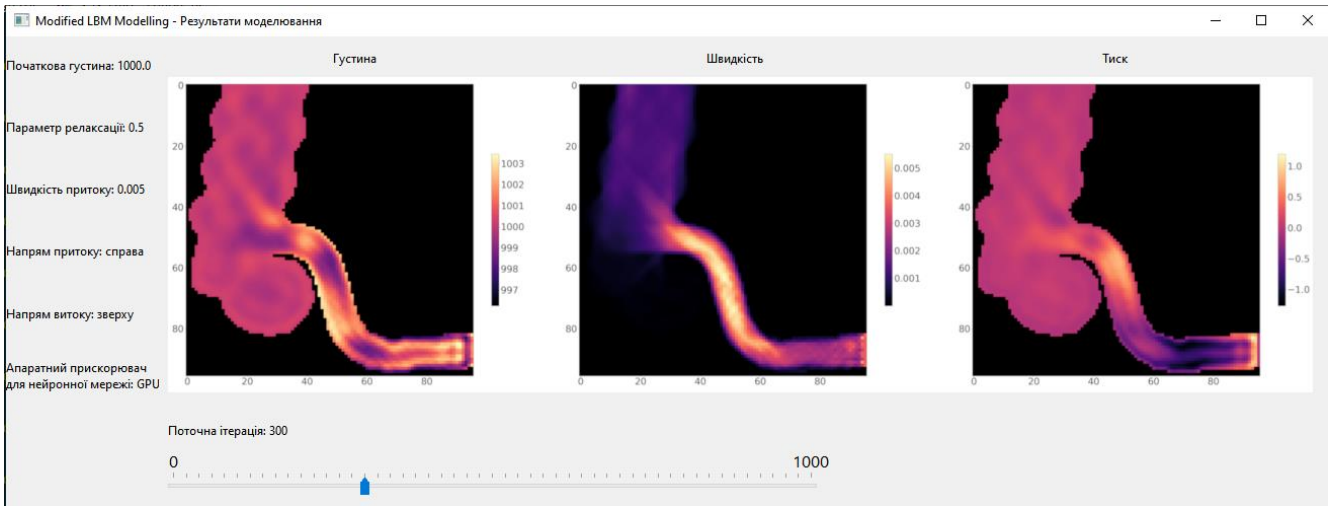

Рисунок 4.20 – Результати моделювання на вибраній ітерації

Розроблене програмне забезпечення дає можливість для базового аналізу результаті моделювання. Для проведення більш ширшого та глибшого аналізу, необхідні більш спеціалізовані інструменти або специфічний програмний код. Але вихідні дані, які є результатом проведеного за допомогою розробленого програмного забезпечення моделювання, мають зручний формат, що є корисним і зручним доповненням до аналітичних програм або скриптів, в тому числі написаних на мові програмування Python.

Результати експериментів, проведених за допомогою розробленого програмного забезпечення показані в розділі 5.

Кодова база програми моделювання руху рідин наведена у додатку А.

4.6 Висновки до розділу 4

В четвертому розділі був розроблений дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі.

Була описана структура нейронної мережі для моделювання розв’язку крайової задачі на основі рівняння Пуассона, особливості генерації тренувального датасету для нейронної мережі, розглянуті особливості приведення результатів роботи нейронної мережі до значень тиску, який використовується для корекції значень поля швидкості в методі LBM. В якості підходу до побудови нейронної мережі був обраний тип нейронної мережі, під назвою FPN. Відповідно до цього підходу, Структура розробленої нейронної мережі складається з двох частин: «bottom-up pathway» і «top-down pathway». Метою частини «bottom-up pathway» є виділення ознак на різних рівнях – від ознак низького рівня, які представляють окрему комірку та її околиці, до особливостей високого рівня, що представляють всю область моделювання. Частина «top-down pathway» поєднує всі отримані на різних рівнях ознаки, що дозволяє отримувати точні рішення. Структура нейронної мережі розроблена з урахуванням геометрії обчислювального простору, де виконується моделювання,

Описаний детальний алгоритм дворівневого методу моделювання руху рідини, де кожна ітерація моделювання реалізує схему предиктора та коректора. Також розроблений метод розпаралелювання для цього методу на основі підходу domain decomposition. Даний паралельний метод є масштабованим для будь-якої квадратної обчислювальної області та довільної кількості процесорів. Крок предиктора в розробленому дворівневому методі був покращений за рахунок модифікованої рівноважної функції на основі на основі мінімізації дискретної ентропії.

З метою зменшення кількості обчислень необхідних для моделювання, були розглянуті види спеціалізованих обчислювальних архітектур для штучних нейронних мереж. На основі їх переваг і недоліків, нейронна мережа була

оптимізована під обраний спеціалізований обчислювальний пристрій NPU Zynq UltraScale+ MPSoC ZCU104.

Для тестування та аналізу ефективності розробленого методу та спеціалізованого прискорювача Zynq UltraScale+ MPSoC ZCU104, було розроблене тестове програмне забезпечення для моделювання руху рідин в довільних обчислювальних просторах розміру 96×96, з можливістю зміни параметрів моделювання, за допомогою мови програмування Python і бібліотек PyQt6, Numpy, matplotlib та PIL.

РОЗДІЛ 5
АНАЛІЗ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ РУХУ РІДИН ДВОРІВНЕВИМ МЕТОДОМ

В даному розділі дисертації були проведені експерименти зі застосуванням розробленого дворівневого методу моделювання руху рідин, який був реалізований у вигляді програмного забезпечення, що описане в четвертому розділі. На основі даних експериментів, була проаналізована точність та стійкість розробленого методу при моделюванні руху рідин в різних обчислювальних областях. Також було порівняна ефективність використання розробленої нейронної мережі для моделювання розв'язку крайової задачі Пуассона, в порівнянні з чисельним методом, де були досліджені точність та швидкість. Були досліджені влив використання апаратних прискорювачів GPU та NPU точність та швидкість роботи нейронної мережі.

5.1 Опис тестових геометрій обчислювального простору

Для проведення експериментів, які моделюють середовища, що наближені до реальних і моделювання яких є актуальним, ми підготували кілька геометрій обчислювального простору, які зображені на рис. 5.1.

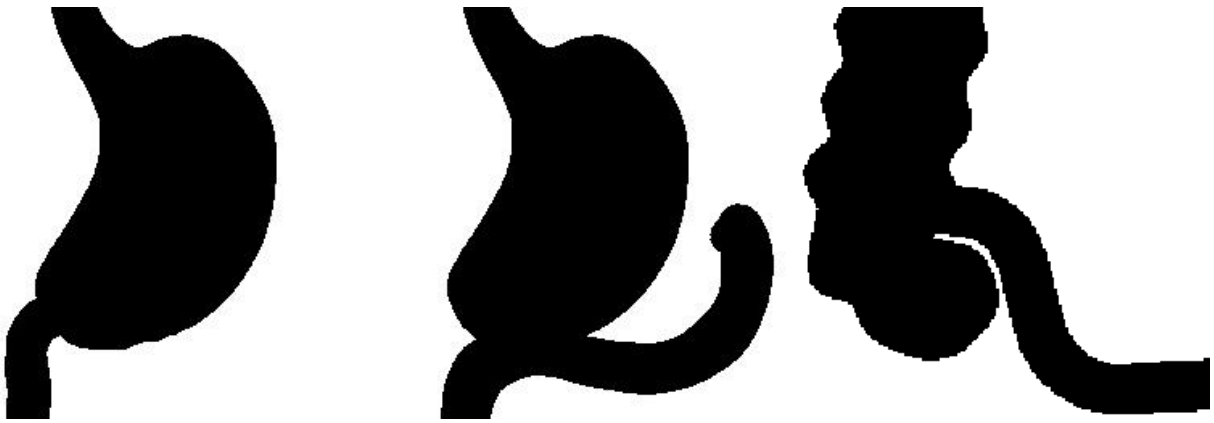

Рисунок 5.1 – Тестові обчислювальні області. Зліва – модель шлунку, по центру – шлунок в стані анастомозу, справа – ділянка сліпої кишки

Розроблені моделі є двовимірними вертикальними перерізами елементів шлунково-кишкового тракту людини. Перша модель – шлунок в нормальному стані. Друга модель – шлунок в стані анастомозу, анастомоз - з'єднання між собою кровоносних чи лімфатичних судин або волокнистих утворів — нервів, м'язів, відділів кишечника та інших порожнистих структур при хірургічних втручаннях [224]. Третя модель – ділянка кишківника, де тонка кишка переходить в товсту. Розроблені моделі є актуальними для біомедичної інженерії.

Далі в тексті поточного розділу ці моделі будуть найменуватись як область №1 (шлунок в нормальному стані), область №2 (шлунок в стані анастомозу) і область №3 (ділянка сліпої кишки).

5.2 Аналіз компенсації стисливості дворівневим методом моделювання руху рідин

З метою дослідження наскільки ефективно розроблений метод компенсує стисливість в порівнянні з класичним методом LBM, була проведена серія експериментів з метою визначення наскільки середня густина рідини в обчислювальній області змінюється під час моделювання в порівнянні з початковим значенням густини.

Для кожної із описаної областей було здійснено моделювання руху рідини. Параметри моделювання наступні: початкова густина $\rho = 1000$, швидкість притоку $v_0 = 0.005$ і два значення параметру релаксації $\tau = 0.5$ і $\tau = 0.7$.

Розглянемо детальніше зміну значень густини рідини під час моделювання кожної області

Область №1

На рис. 5.2 та 5.3 зображена зміна середньої густини рідини в обчислювальній області при параметрах релаксації $\tau = 0.5$ і $\tau = 0.7$ відповідно, при застосуванні розробленого методу та звичайного методу LBM.

Результати експериментів свідчать про здатність розробленого методу компенсувати ефекти стисливості, в порівнянні зі звичайним методом LBM при

параметрі релаксації $\tau = 0.5$. При моделюванні параметру $\tau = 0.7$ в обох методах спостерігається збільшення середньої густини рідини, але при використанні розробленого методу це збільшення відбувається з меншою швидкістю.

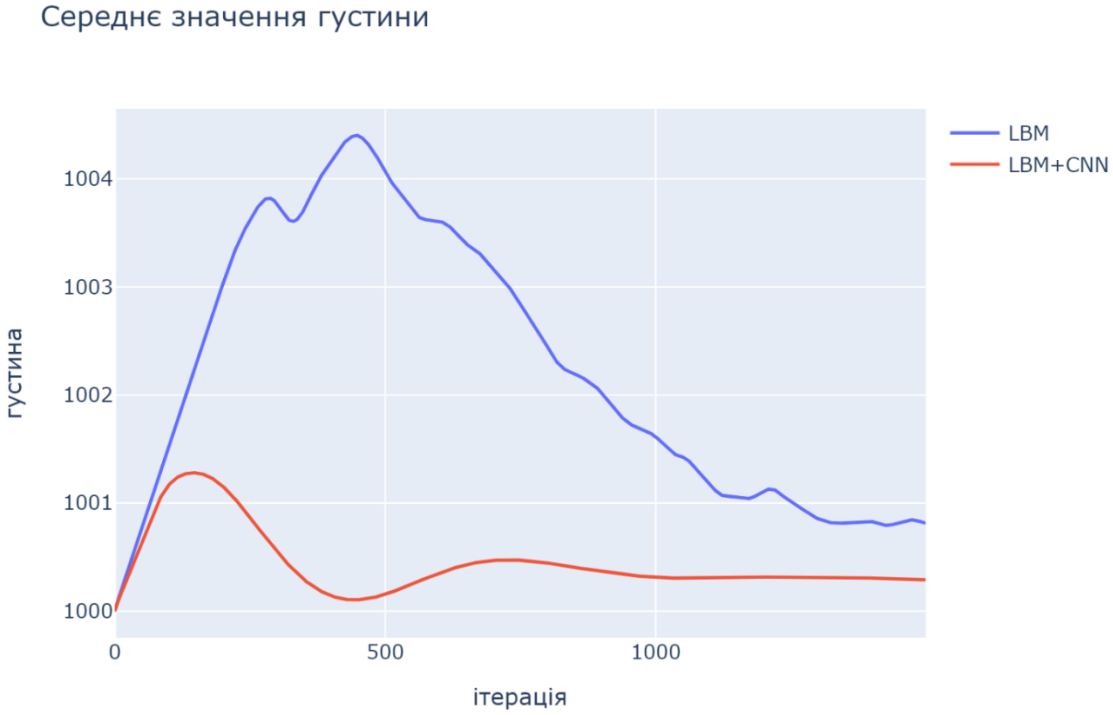

Рисунок 5.2 – Зміна середнього значення густини при $\tau = 0.5$ під час моделювання області №1

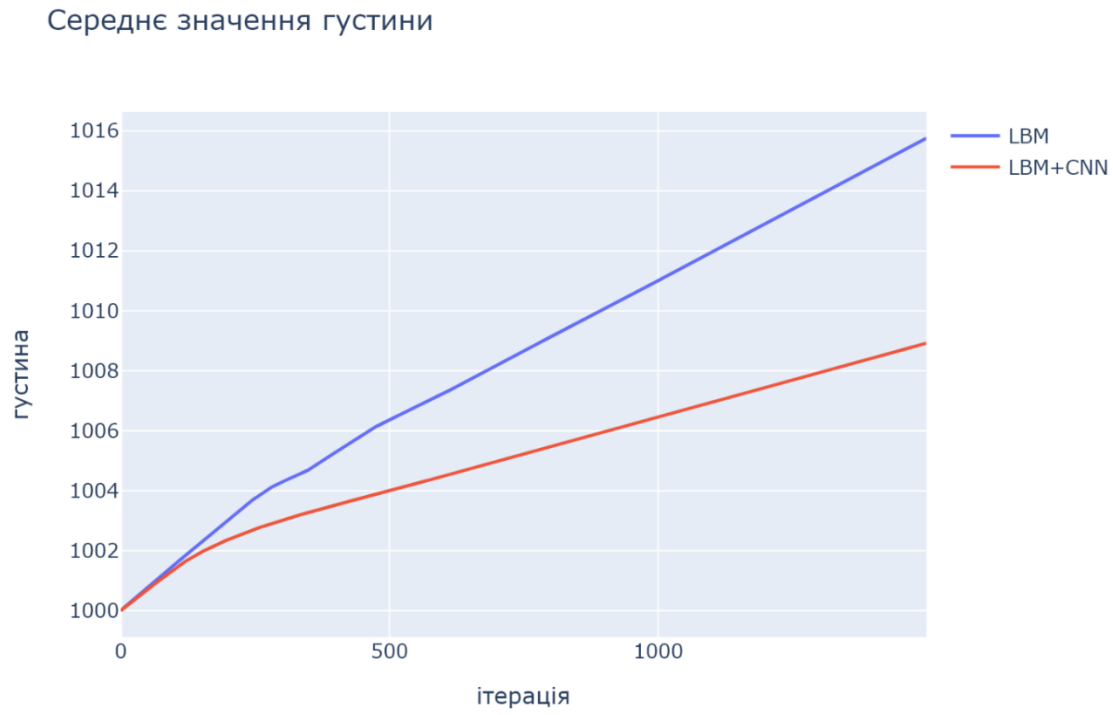

Рисунок 5.3 – Зміна середнього значення густини при $\tau = 0.7$ під час моделювання області №1

Область №2

На рис. 5.4 та 5.5 зображена зміна середньої густини рідини в обчислювальній області при параметрах релаксації $\tau = 0.5$ і $\tau = 0.7$ відповідно, при застосуванні розробленого методу та звичайного методу LBM.

Як і у випадку області №1, результати експериментів свідчать про здатність розробленого методу компенсувати ефекти стисливості, в порівнянні зі звичайним методом LBM при параметрі релаксації $\tau = 0.5$. Аналогічно, при моделюванні параметру $\tau = 0.7$ в обох методах спостерігається збільшення середньої густини рідини, але при використанні розробленого методу це збільшення відбувається з меншою швидкістю.

Область №3

На рис. 5.6 та 5.7 зображена зміна середньої густини рідини в обчислювальній області при параметрах релаксації $\tau = 0.5$ і $\tau = 0.7$ відповідно, при застосуванні розробленого методу та звичайного методу LBM.

Як і у випадку областей №1 і №2, результати експериментів свідчать про здатність розробленого методу компенсувати ефекти стисливості, в порівнянні зі звичайним методом LBM при параметрі релаксації $\tau = 0.5$. Аналогічно, при моделюванні параметру $\tau = 0.7$ в обох методах спостерігається збільшення середньої густини рідини, але при використанні розробленого методу це збільшення відбувається з меншою швидкістю.

Таким чином у всіх проведених експериментах розроблений метод показав кращу здатність забезпечувати нестисливість модельованої рідини.

5.3 Дослідження обчислювальної ефективності дворівневого методу

Щоб дослідити і проаналізувати швидкість моделювання розробленого методу, були виміряний середній час моделювання однієї ітерації різних методів моделювання. Також був виміряний час моделювання в залежності від апаратного прискорювача, який забезпечував роботу нейронної мережі.

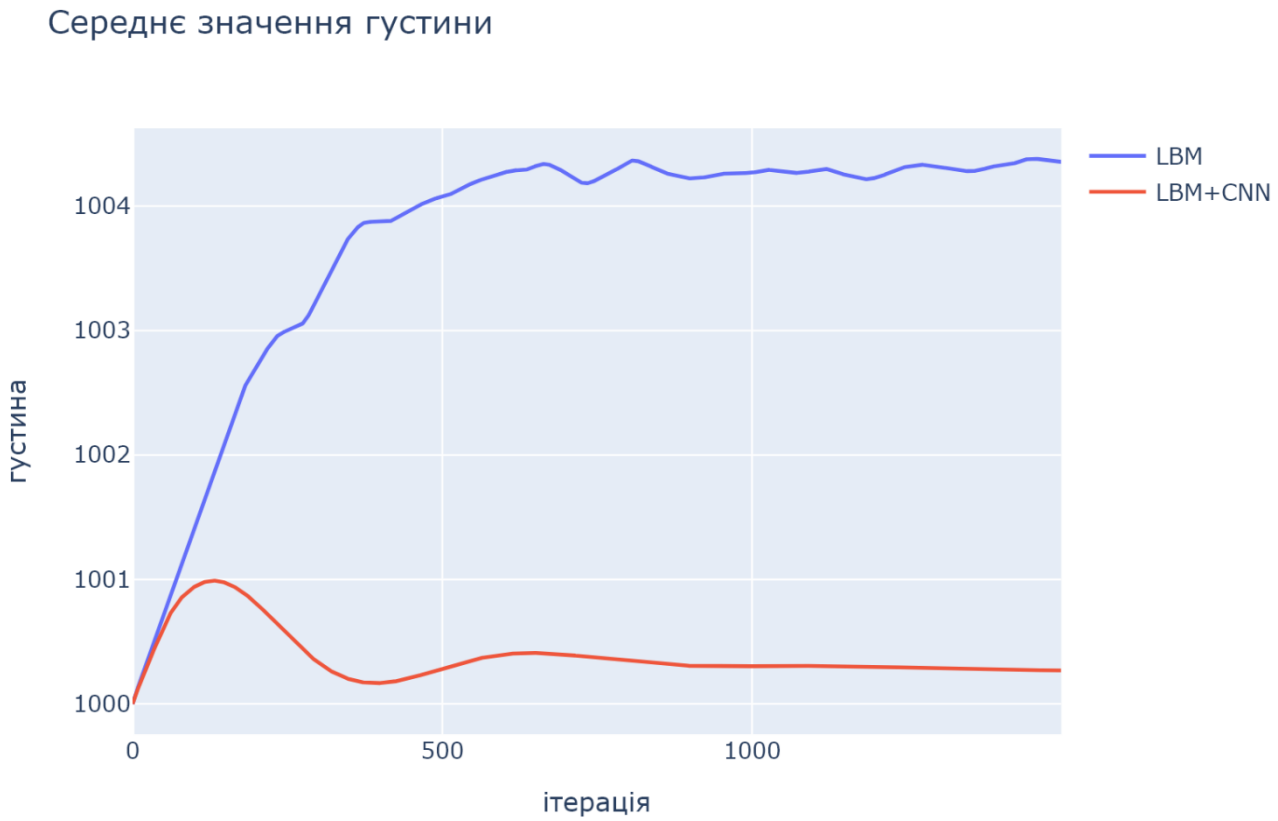

Рисунок 5.4 – Зміна середнього значення густини при $\tau = 0.5$ під час моделювання області №2

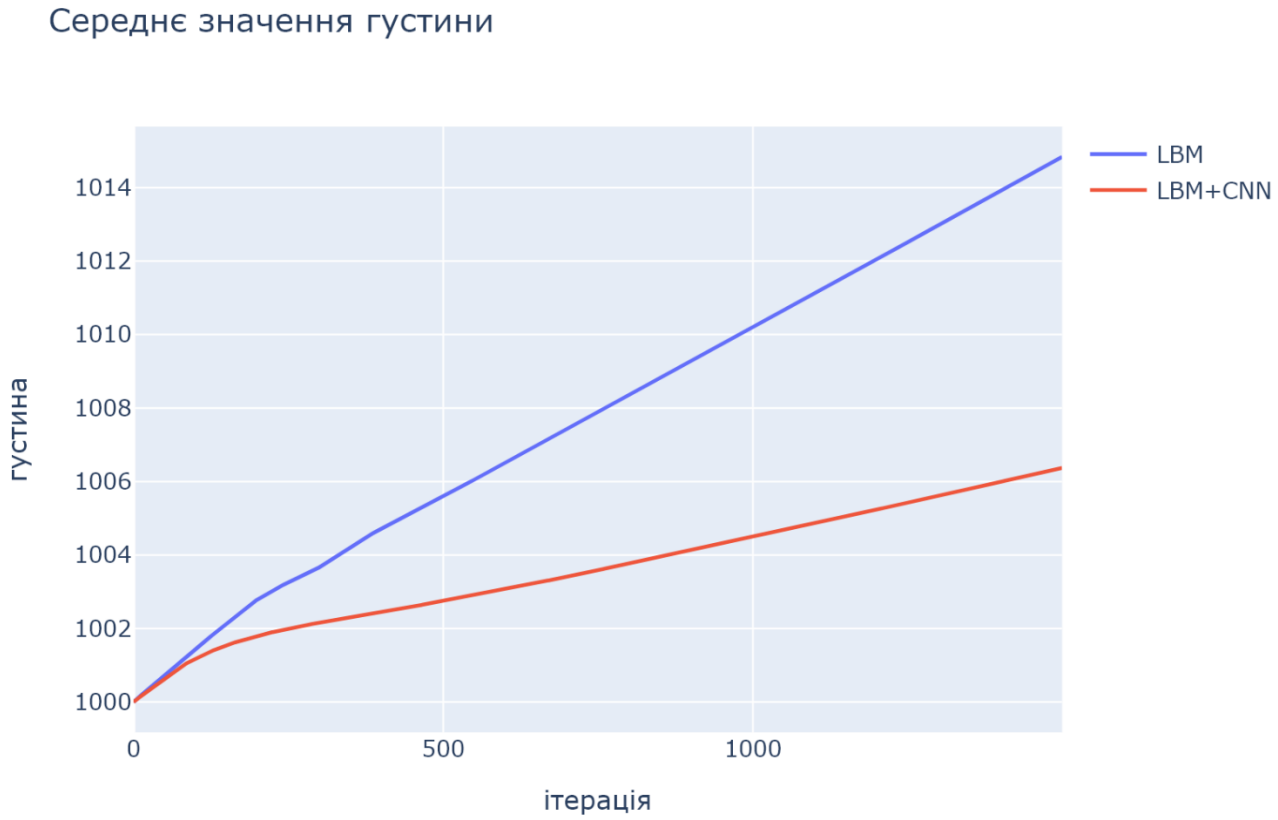

Рисунок 5.5 – Зміна середнього значення густини при $\tau = 0.7$ під час моделювання області №2

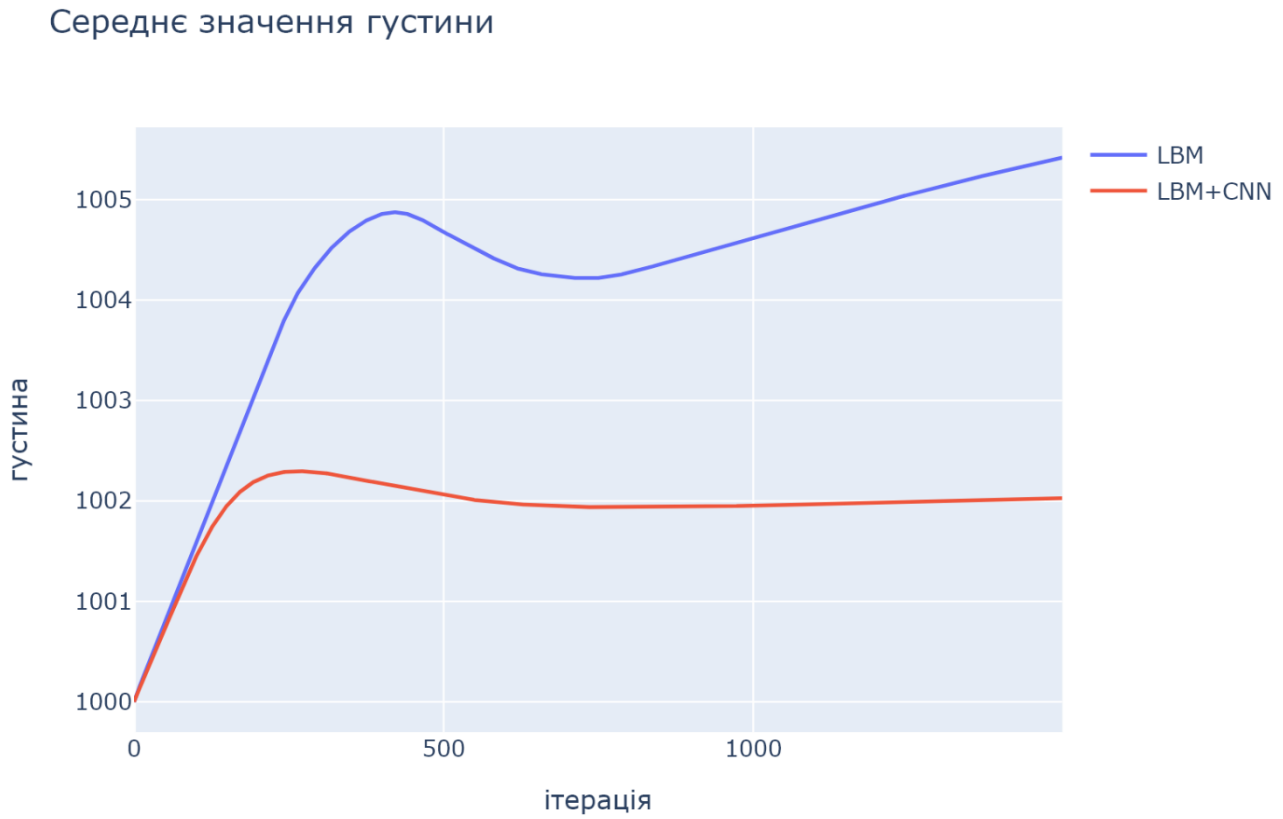

Рисунок 5.6 – Зміна середнього значення густини при $\tau = 0.5$ під час моделювання області №3

Рисунок 5.7 – Зміна середнього значення густини при $\tau = 0.7$ під час моделювання області №3

Так, на описаних вище областях №1,2 і 3 були проведені експерименти такими методами:

1. Звичайний метод LBM
2. Дворівневий метод LBM з використанням чисельного методу Якобі для вирішення рівняння Пуассона для тиску
3. Дворівневий метод LBM з використанням розробленої нейронної мережі для вирішення рівняння Пуассона для тиску та графічним процесором GPU
4. Дворівневий метод LBM з використанням розробленої нейронної мережі для вирішення рівняння Пуассона для тиску та апаратним прискорювачем NPU, описаний в розділі 4.4

Результати експериментів показані в таблиці 5.1

Таблиця 5.1 – Середній час моделювання однієї ітерації руху рідини

	Час моделювання однієї ітерації, с			
	LBM	LBM+чисельний метод	LBM+нейронна мережа+GPU	LBM+нейронна мережа+NPU
Область №1	0.00194	0.23102	0.03813	0.01678
Область №2	0.00187	0.22986	0.03794	0.01731
Область №3	0.00174	0.23231	0.03831	0.01763

Проаналізувавши результати експериментів, виводяться наступні висновки:

- Звичайний метод LBM швидший за розроблений дворівневий метод, що є очевидним, оскільки дворівневий метод включає додатковий крок у вигляді моделювання розв’язку рівняння Пуассона для тиску
- Використання нейронної мережі значно пришвидшує розроблений дворівневий метод, в порівнянні з використанням чисельного метода для

розв’язання рівняння Пуассона. Прискорення сягає від 6,05 до 13,76 разів, в залежності від типу апаратного прискорювача для нейронної мережі

- NPU дає прискорення розробленого методу в порівнянні з GPU від 2,17 до 2,27 разів.

5.4 Аналіз поля швидкості руху рідини

Було проаналізовано значення модулю швидкості рідини в обчислювальній області при різних параметрах моделювання з метою порівняння стабільності та точності розробленого методу в порівнянні зі звичайним методом LBM. Були використані ті ж самі параметри і обчислювальні області, що й під час дослідження компенсації стисливості. Заміри значень швидкості були здійснені на 500 та 1000 ітераціях моделювання.

Область №1

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.5$ зображені на рис. 5.8 і 5.9. Значення швидкості рідини в ділянках притоку і витоку при використанні розробленого методу на 20-30% менші, ніж при використанні звичайного методу LBM. Також за допомогою візуалізації, можна відмітити, що при застосуванні дворівневого методу розподіл поля швидкості є

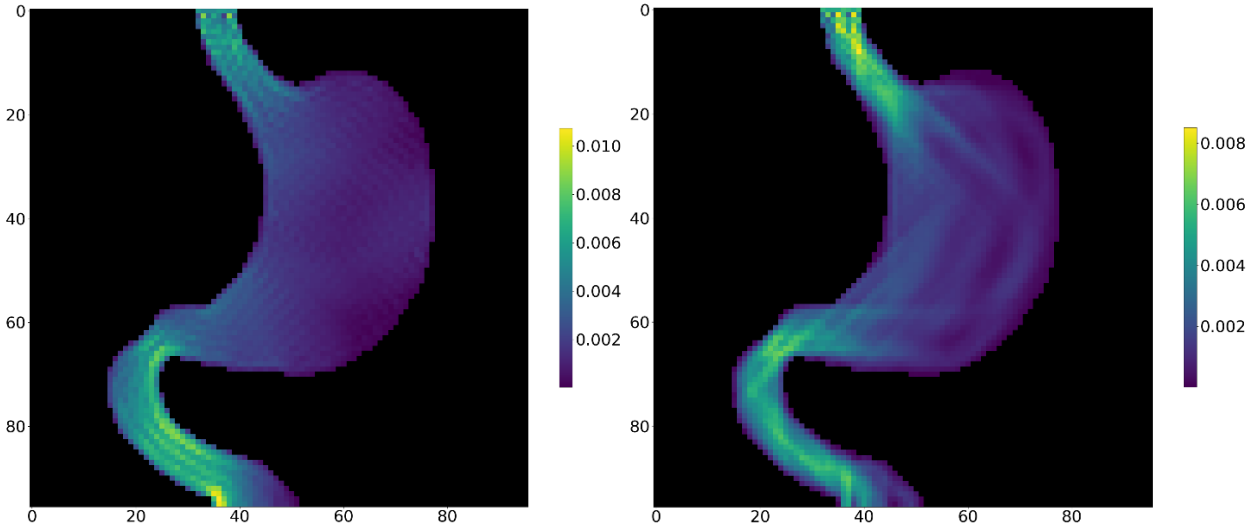

Рисунок 5.8 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

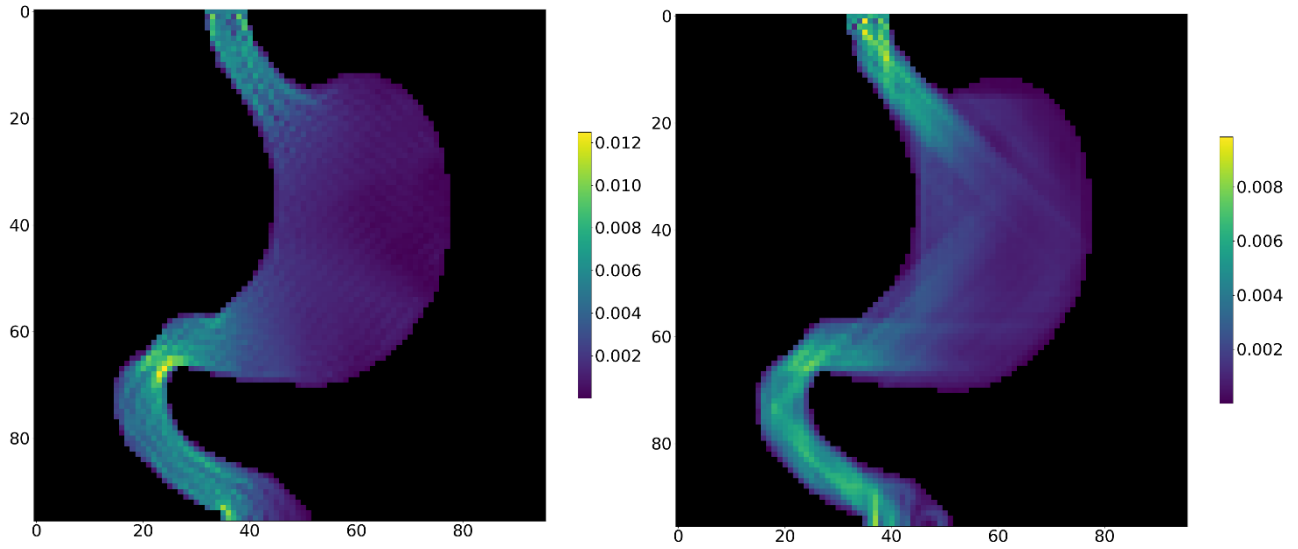

Рисунок 5.9 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

більш нерівномірним і можна виділити окремі потоки рідини. Під час використання звичайного методу LBM, це зробити значно тяжче.

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.7$ зображені на рис. 5.10 і 5.11. В даному випадку значення полей швидкості не надто відрізняється в обох випадках. Варто відзначити, що швидкість рідини в області витоку при використанні дворівневого методу є більшою, ніж при використанні звичайного методу LBM.

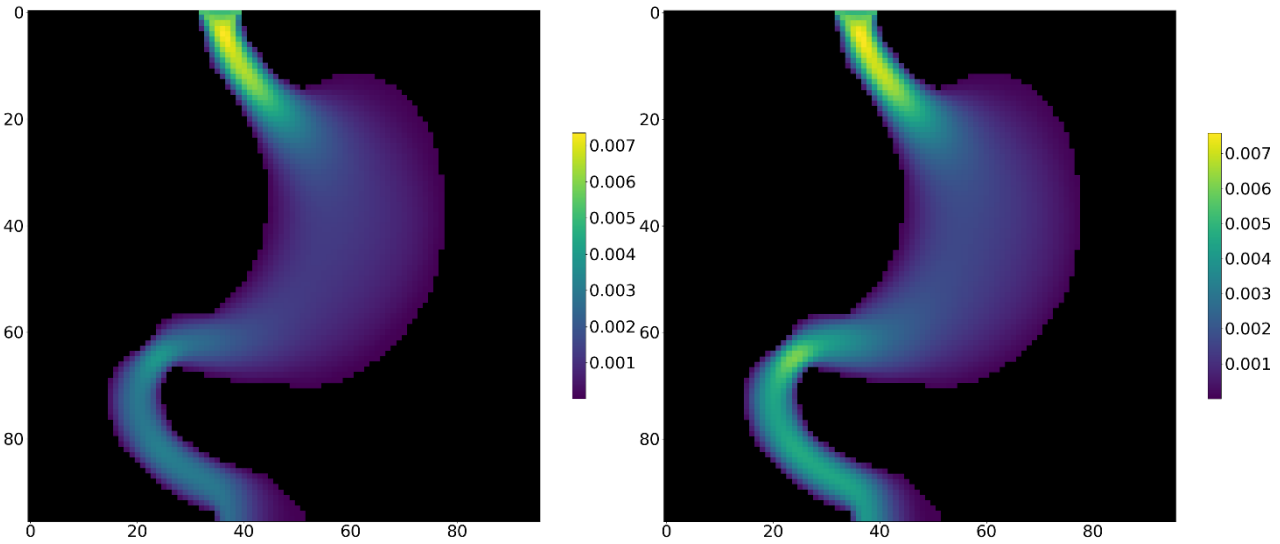

Рисунок 5.10 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

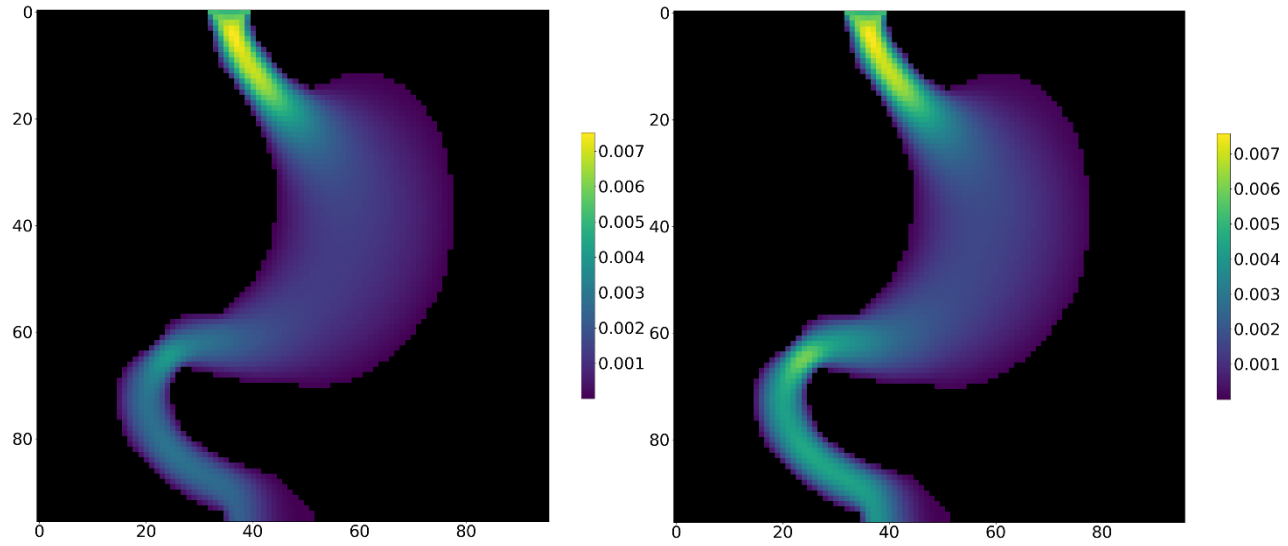

Рисунок 5.11 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

Область №2

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.5$ зображені на рис. 5.12 і 5.13. Як і у випадку з областю №1, можна виділити окремі потоки рідини при застосуванні дворівневого методу, оскільки розподіл поля швидкості є більш нерівномірним. На 1000-ій ітерації значення швидкості рідини в ділянках притоку і витоку при використанні розробленого методу на 10-20% менші.

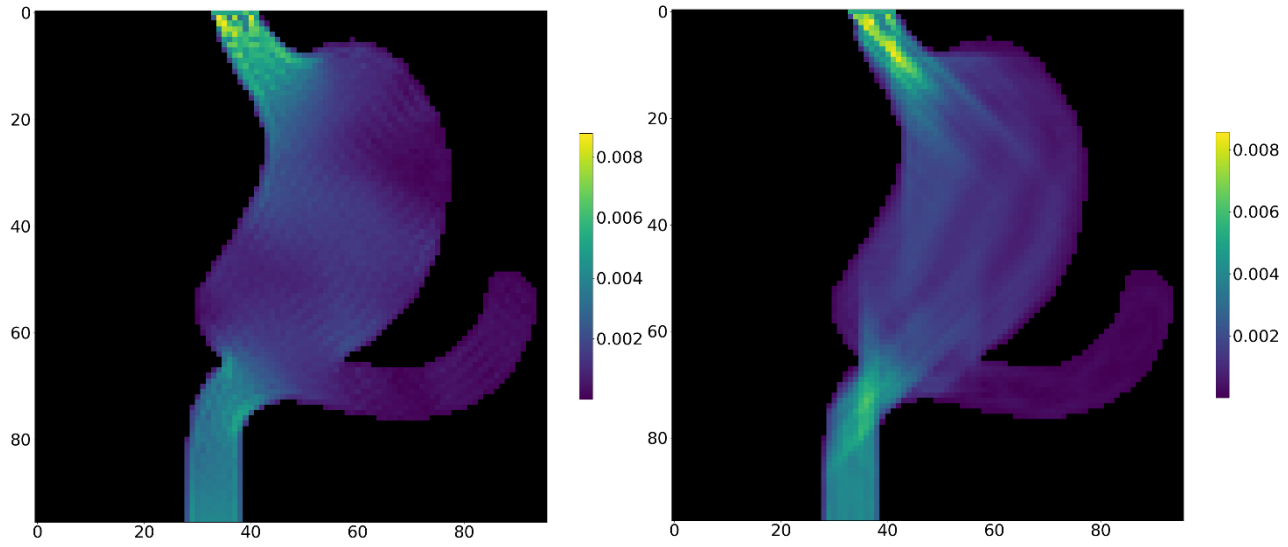

Рисунок 5.12 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

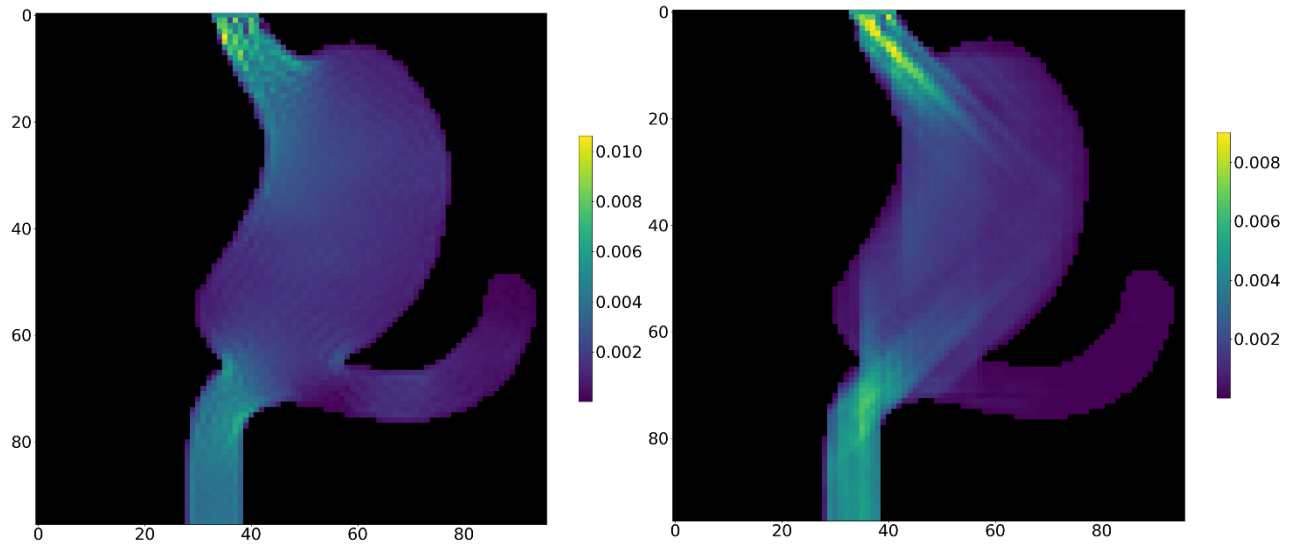

Рисунок 5.13 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.7$ зображені на рис. 5.14 і 5.15. Як і у випадку з областю №1, значення полей швидкості не надто відрізняється при застосуванні обох методів, і швидкість рідини в області витоку при використанні дворівневого методу є більшою, ніж при використанні звичайного методу LBM.

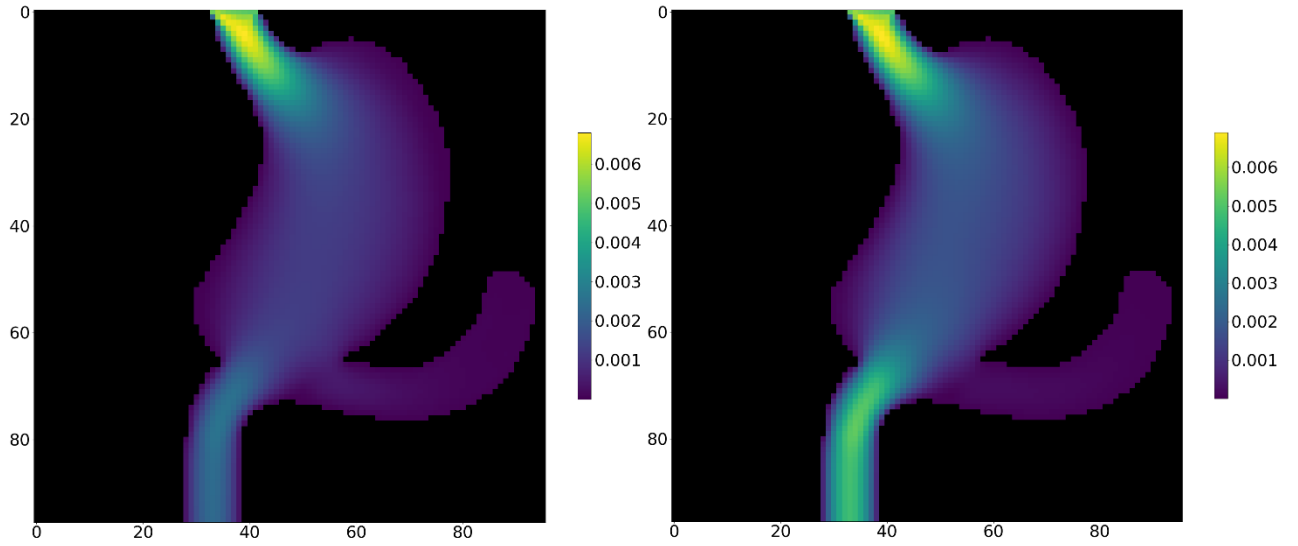

Рисунок 5.14 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

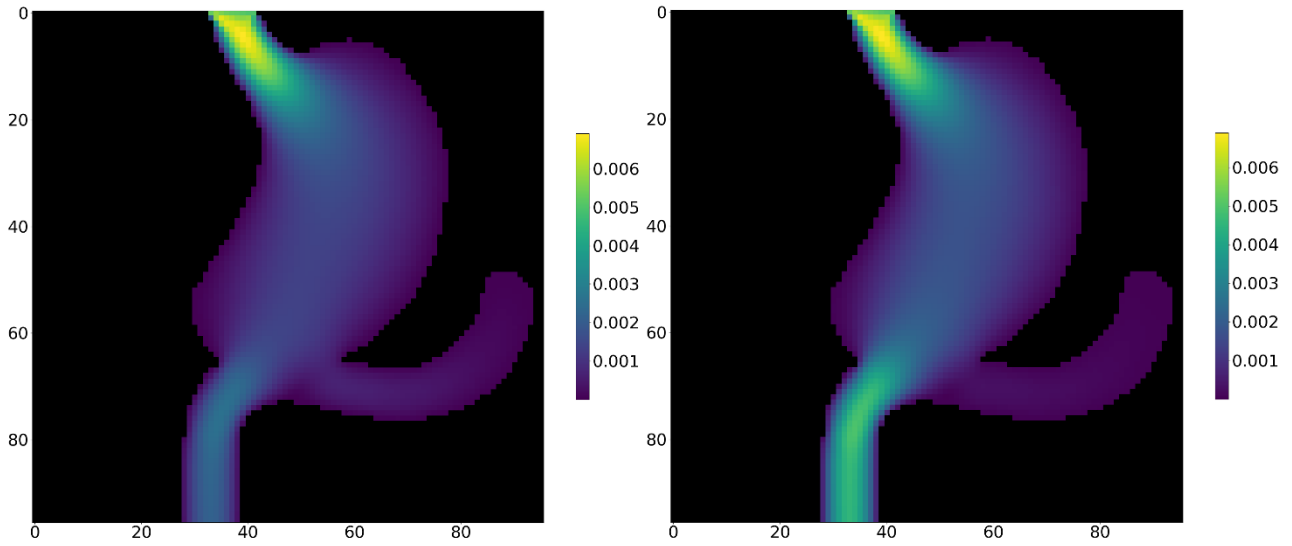

Рисунок 5.15 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

Область №3

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.5$ зображені на рис. 5.16 і 5.17. Аналогічно з моделюванням областей №1 та №2, можна виділити окремі потоки рідини при застосуванні дворівневого методу, оскільки розподіл поля швидкості є більш нерівномірним. Аналогічно, на 1000-ій ітерації значення швидкості рідини в ділянках притоку і витоку при використанні розробленого методу на 10-20% менші.

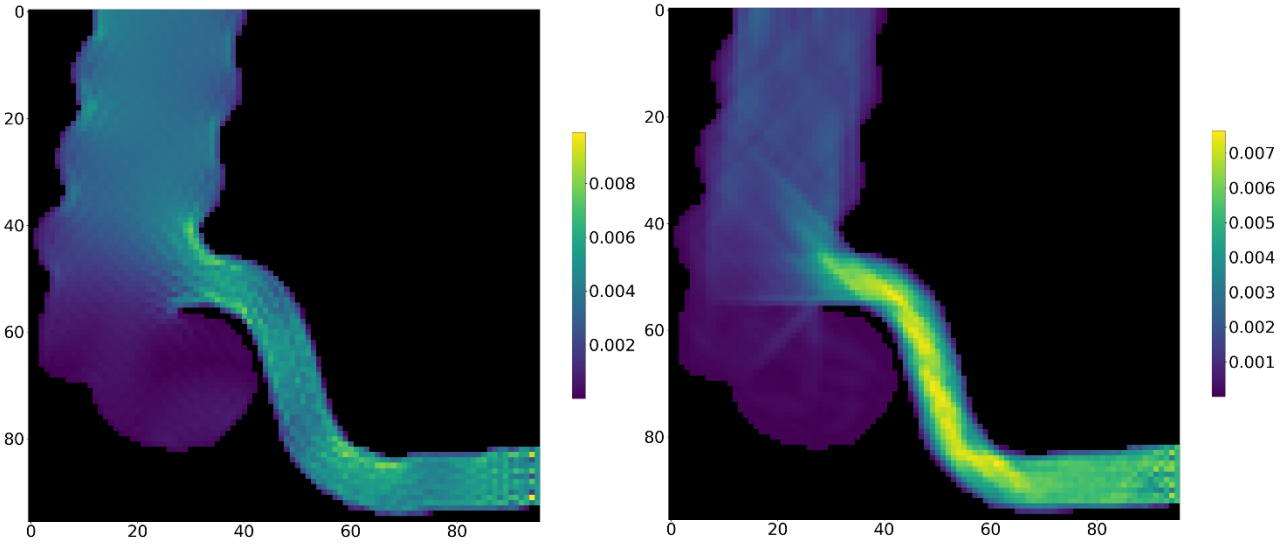

Рисунок 5.16 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

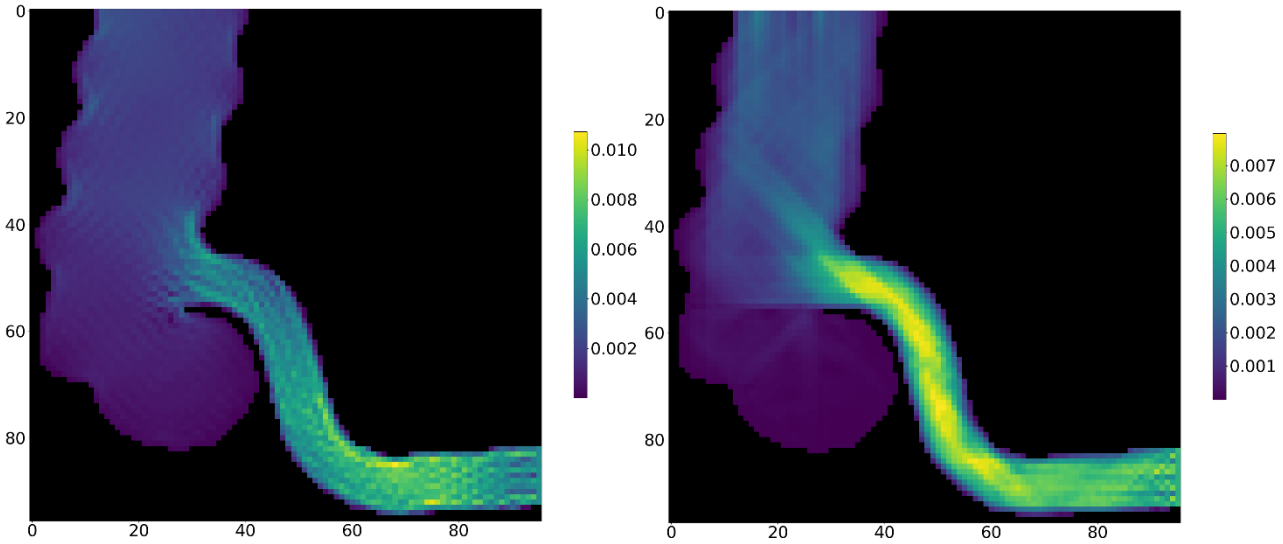

Рисунок 5.17 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.5$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

Розподіли полів швидкостей у досліджуваних методах при параметрі релаксації $\tau = 0.7$ зображені на рис. 5.18 і 5.19. . Як і у випадку з областями №1 та №2, значення полей швидкості не надто відрізняється при застосуванні обох методів. Але на відміну, від попередніх областей, значення швидкості рідини в області витоку незначно відрізняються при застосування обох методів. Це можна пояснити більшою шириною ділянки витоку рідини відносно ширини ділянки притоку.

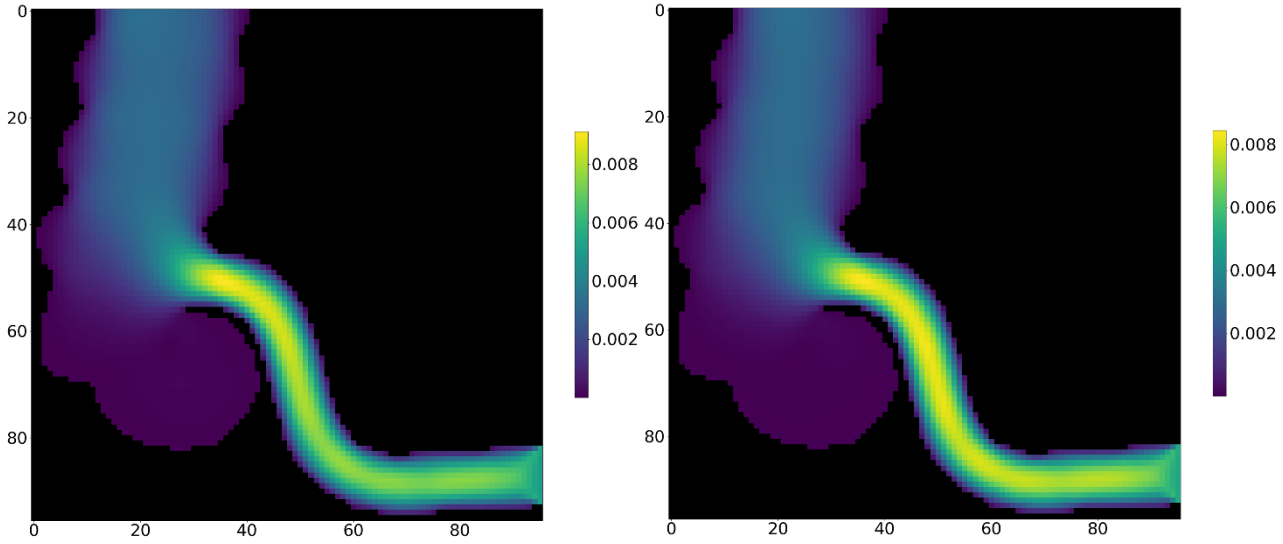

Рисунок 5.18 – Розподіл поля швидкостей на 500-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

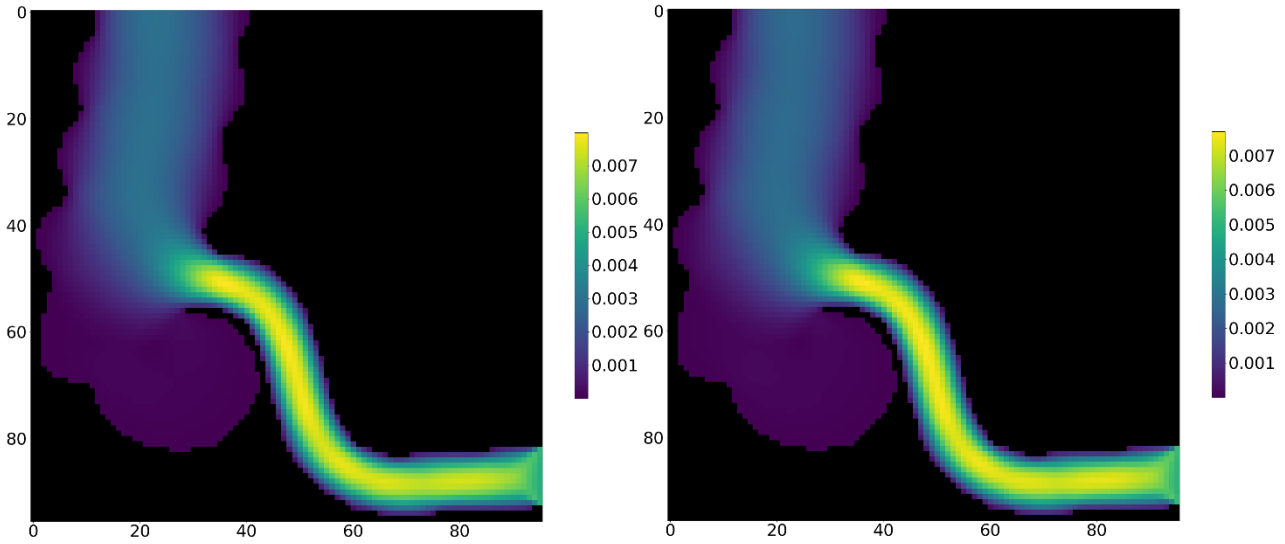

Рисунок 5.19 – Розподіл поля швидкостей на 1000-ій ітерації, $\tau = 0.7$. Зліва – звичайний метод LBM, справа – запропонований дворівневий метод

5.5 Моделювання течії в горизонтальній трубці

Розроблений метод було протестовано для однієї із поширених класичних задач в обчислювальній гідродинаміці – моделюванні течії Пуазейля [233], або ламінарної течії в плоскій трубці. Схема руху рідини зображена на рис. 5.20.

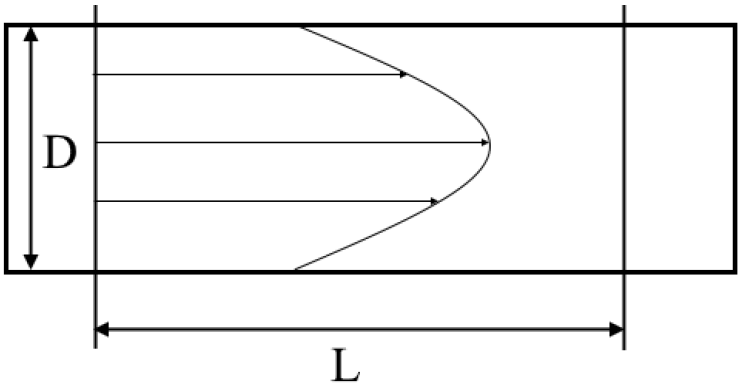

Рисунок 5.20 – Схему руху рідини в трубці

Ми розробили модель перерізу трубки з відношенням довжини до висоти, що дорівнює 3. Зверху і знизу задали граничну умову твердої стінки, ліворуч – умову притоку рідини, праворуч – витоку рідини. Для оцінки точності симуляції розробленої моделі було промодельовано рух ламінарної течії з числами Рейнольдса 100 та 1000. Після 2000 ітерацій моделювання було порівняно профілі

швидкості на правому краї трубки з аналітичним розв'язком. Результати моделювання представлені на рис. 5.21 і 5.22.

При значенні числа Рейнольдса 100 чітко видно відповідність розробленого дворівневого методу моделювання аналітичному розв'язку. Максимальне значення відносної похибки становить 0.8%. При значенні числа Рейнольдса 1000 зберігається згадана вище відповідність, точність стала гіршою – відносна похибка становить 1.7%.

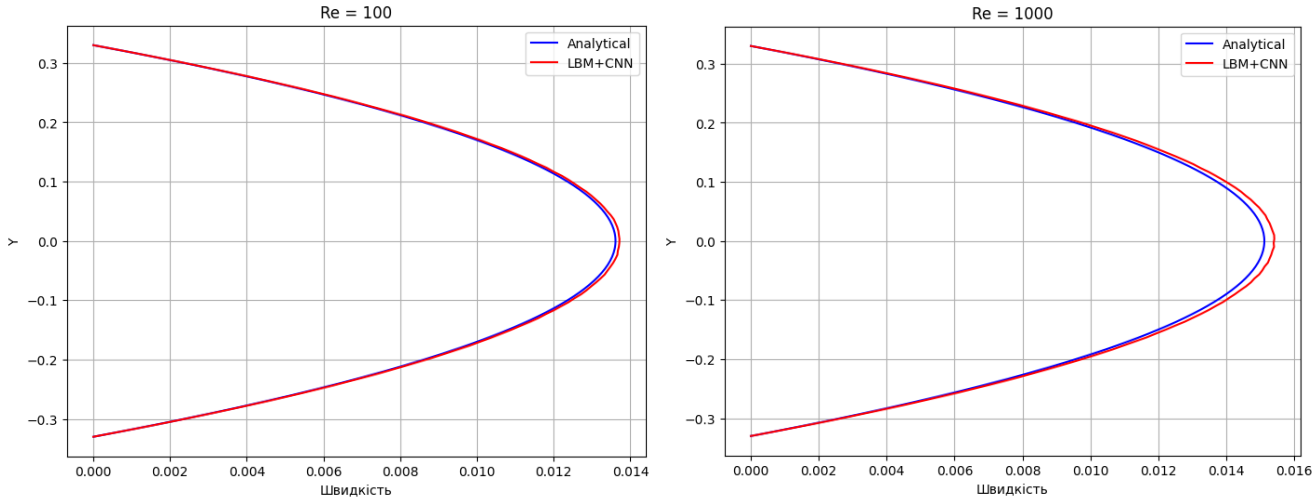

Рисунок 5.21 – Профілі швидкості руху рідини, що змодельовані аналітичним способом та розробленим дворівневим методом, Re = 100 (зліва) та Re=1000 (справа)

5.6 Висновки до розділу 5

В даному розділі був проведений аналіз результатів моделювання руху рідин за допомогою розробленого дворівневого метод моделювання руху рідин, що включає в себе модифікований метод LBM та згорткову нейронну мережу для моделювання розв'язку крайової задачі на основі рівняння Пуассона.

Для проведення експериментів моделювання середовищ, моделі яких наближені до реальних, були підготовлені кілька геометрій обчислювального простору, які є двовимірними вертикальними перерізами елементів шлунково-кишкового тракту людини.

Була проведена серія експериментів з метою визначення наскільки середня густина рідини в обчислювальній області змінюється під час моделювання в

порівнянні з початковим значенням густини. Результати експериментів підтвердили здатність розробленого дворівневого методу забезпечувати нестисливість рідини, в порівнянні зі звичайним методом LBM. Була досліджена точність нейронної мережі в порівнянні з чисельним методом. Була показана взаємна відповідність між ними. Також була досліджена точність моделювання руху ламінарного потоку в трубці, шляхом порівняння значення профілю швидкості в порівнянні з аналітичним розв’язком. Експеримент показав незначну похибку обчислень. Це свідчить про точність та надійність розробленого методу.

Також була досліджена обчислювальна швидкість розробленого методу та вплив використання різних апаратних прискорювачів на кількість обчислень, в порівнянні з іншими методами моделювання руху рідин. В якості міри порівняння був обраний середній час моделювання однієї ітерації кожним методом. Так, дворівневий метод з використанням нейронної мережі, показав більш ніж у 6 разів кращу ефективність, ніж з використанням чисельного методу Якобі, при використанні GPU, та у 13 разів кращу ефективність, при використанні NPU.

ВИСНОВКИ

В дисертаційній роботі вирішено актуальне наукове завдання пришвидшення математичного моделювання руху нестисливих рідин за допомогою моделі Больцмана та згорткової нейронної мережі.

Основні результати виконаної роботи:

1. Проведений огляд та аналіз методів моделювання динаміки рідини, досліджені основні типи підходів до моделювання руху рідини.
2. Обгрунтована можливість застосування методу LBM для моделювання руху рідин. Для досягнення безумовної лінійної стабільності моделювання була описана модифікована функція розподілу на основі мінімізації дискретної ентропії. Вона була використана для забезпечення безумовної лінійної стабільності моделювання.
3. Обгрунтована необхідність уточнення поля швидкості за допомогою рівняння Пуассона для тиску під час моделювання руху нестисливих рідин методом LBM. Сформульована крайова задача на основі рівняння Пуассона для тиску.
4. Розроблений дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі, що базується на корекції значень поля швидкості на основі значень тиску, що обчислюються за допомогою моделювання розв’язку крайової задачі на основі рівняння Пуассона для тиску.
5. Розроблена структура нейронної мережі для моделювання розв’язку крайової задачі на основі рівняння Пуассона, та здійснено її тренування. Описані деталі та особливості тренування нейронної мережі. Здійснена адаптація розробленої нейронної мережі під спеціальний обчислювальний пристрій NPU.
6. Розроблений паралельний масштабований метод для дворівневого методу моделювання руху рідин на основі підходу domain decomposition, для

будь-якої квадратної обчислювальної області та довільної кількості процесорів.

7. Розроблене програмне забезпечення яке реалізувало розроблений дворівневий метод моделювання руху рідин. Можливості програми включаються в себе: задавання параметрів течії рідини, завантаження геометрій обчислювального простору, збереження даних моделювання на пам’ять, відтворення результатів моделювання.

8. Проведений аналіз результатів моделювання руху рідин за допомогою розробленого дворівневого методу моделювання руху рідин. За результатами експериментів була підтверджена здатність розробленого дворівневого методу забезпечувати нестисливість рідини, в порівнянні зі звичайним методом LBM. Була досліджена точність нейронної мережі в порівнянні з чисельним методом, яка показала взаємну відповідність між ними. досліджена обчислювальна швидкість розробленого методу та вплив використання різних апаратних прискорювачів на кількість обчислень. Так, дворівневий метод показав більш ніж у 6 разів кращу ефективність, ніж чисельний метод, при використанні GPU, та у 13 разів кращу ефективність, при використанні NPU.

ЛІТЕРАТУРА

1. Chorin, A. J. (1967) A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.,2(1),* pp. 12–26.

2. Лойцянский, Л. Г. (1978) Механика жидкости и газа. М.: Наука, 736.

3. Шлихтинг, Г. (1969) Теория пограничного слоя. М.: Наука, 742.

4. Harlow, F. H., Welch, J. E. (1965) Numerical calculation of time-dependent viscouse incompressible flow of fluid with free surface. *Phys. Fluids*, *8(12),* pp. 2182–2189.

5. Chorin, A.J. (1968) Numerical solution of the Navier-Stokes equations, *Math. Comp.*, 22, pp. 745-762.

6. Temam, R. (1969) Sur l'approximation de la solution des equations de Navier-Stokes par la m´ethode des fractionnarires II, *Arch. Rational Mech. Anal.*, 33, pp. 377–385.

7. Douglas, J., Gunn, J. E. (1964) A general formulation of alternating direction implicit methods, *Numer. Math.*,6, pp. 428–453.

8. Peaceman, D. W., Rachford, H. H. (1955) The numerical solution of parabolic and elliptic differentional equations, *J. Soc. Indust. Appl. Math.* 3, pp. 28–41.

9. Hundsdorfer, W., Verwer, J. G. (2003) Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations, *Springer*.

10. Mayne, D. Q., Reinelt, G. (1971) Alternating direction methods for parabolic equations, *Numerische Mathematik*, 16(4), pp. 303-313.

11. Белоцерковский, О. М., Давыдов, Ю. М. (1971) Нестационарный метод "крупных частиц"для газодинамических расчетов, *Ж. Вычисл. Матем. И матем.Физ*., 11(1), pp. 182–207

12. Белоцерковский, О. М. (1994) *Численное моделирование в механике сплошных сред: 2-е изд., перераб. и доп.*, М.: Физматлит, 448.

13. Patankar, S. V. (1980). *Numerical Heat Transfer and Fluid Flow*. CRC Press.

14. Ferziger, J. H., & Perić, M. (2012). Computational Methods for Fluid Dynamics, *Springer Science & Business Media*.

15. Anderson, J. D. (1995). *Computational Fluid Dynamics: The Basics with Applications*, McGraw-Hill Education.

16. Liu, Xunliang & Tao, Wen-Quan & He, Y.L.. (2005). A simple method for improving the SIMPLER algorithm for numerical simulations of incompressible fluid flow and heat transfer problems, *Engineering Computations*, 22, pp. 921-939.

17. Patankar, S.V. (1981), A calculation procedure for two-dimensional elliptic situations, *Numer. Heat Transfer*, Vol. 4, pp. 409-25.

18. Raithby, G.D., Schneider, G.E. (1988) Elliptic system: finite difference method II, in Minkowycz, W.J., Sparrow, E.M., Pletcher, R.H. and Schneider, G.E., *Handbook of Numerical Heat Transfer*, Wiley, New York, NY, pp. 241-89.

19. Xiang, T.-R., Yang, X.I.A., Shi, Y.-P. (2021) Neuroevolution-enabled adaptation of the Jacobi method for Poisson's equation with density discontinuities, *Theoretical and Applied Mechanics Letters*, 11(3)

20. Ankita, M., Krishna, S.. (2020). Evaluation of Scheduled Relaxation Jacobi Method as Solver and Preconditioner for Numerical Solution of Pressure Poisson Equation.

21. Mohammad Shafaet Islam, Qiqi Wang. (2020). Hierarchical Jacobi Iteration for Structured Matrices on GPUs using Shared Memory. *arXiv preprint arXiv: 2006.16465.*

22. Hockney, R. W. (1965) A Fast Direct Solution of Poisson's Equation Using Fourier Analysis, *Journal of the ACM*, 12(1), pp. 95–113.

23. George, J. A. (1970) The use of direct methods for the solution of the discrete Poisson equation on nonrectangular regions, *Computer Science Rep*, pp. 70-159.

24. Wilhelmson, R. B., Ericksen, J. H. (1977). Direct solutions for Poisson's equation in three dimensions, *Journal of Computational Physics*, 25(4), pp. 319–331.

25. Karelius, F. (2017) Stationary iterative methods : Five methods and illustrative examples. Dissertation. Available at: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-69711.

26. Seidel, L. (1874) "Über ein Verfahren, die Gleichungen, auf welche die Methode der kleinsten Quadrate führt, sowie lineäre Gleichungen überhaupt, durch successive Annäherung aufzulösen" [On a process for solving by successive approximation the equations to which the method of least squares leads as well as linear equations generally]. *Abhandlungen der Mathematisch-Physikalischen Klasse der Königlich Bayerischen Akademie der Wissenschaften* (in German). 11 (3), pp. 81–108.

27. Bara, S., Zimmermann, J. (1999) Ultra-fast Poisson's equation solvers using wired-up processors for virtual devices architectures, *CAS '99 Proceedings, International Semiconductor Conference*

28. Liu, Y., Gao, F., Zhang, L. (2010) Multigrid Method with Adaptive Gauss-Seidel Smoother for Solving Poisson Equations, *2010 2nd International Conference on Information Engineering and Computer Science.*

29. Liebmann, H. (1918), Sitzungsberichte der Bayer, *Akad. Wiss Math.-Phys. Ki.*, 47, 385.

30. Young, D. M. (1989). A historical overview of iterative methods. *Computer Physics Communications*, 53(1), pp. 1–17

31. Fox, L. (1947) Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, *Proc. R. Soc. Lond.*, 190, pp. 31–59.

32. Shortley, G., Weller, R., Fried, B. (1940). *Numerical solution of Laplace's and Poisson's equations with applications to photoelasticity and torsion*. Columbus, College of Engineering, the Ohio State University.

33. Quarteroni, A., Sacco, R., Saleri, F. (2007) Iterative Methods for Solving Linear Systems. In Numerical Mathematics. *Texts in Applied Mathematics Springer, New York, NY*, 37(4), pp. 125–180.

34. Bertaccini, D. Durastante, F. (2018) *Iterative methods and preconditioning for large and sparse linear systems with applications*, Taylor & Francis Group.

35. Mazumder, S. (2016) *Numerical Methods for Partial Differential Equations*, Academic Press, pp. 169–218.

36. Young, D. (1954). Iterative Methods for Solving Partial Difference Equations of Elliptic Type. *Transactions of the American Mathematical Society*, 76(1), pp. 92–111.

37. Yang, S., Gobbert, M. K. (2009). The optimal relaxation parameter for the SOR method applied to the Poisson equation in any space dimensions. *Applied Mathematics Letters*, 22(3), pp. 325–331.

38. Vanderbauwhede, W., Takemi, T. (2015). Twinned buffering: A simple and highly effective scheme for parallelization of Successive Over-Relaxation on GPUs and other accelerators. *2015 International Conference on High Performance Computing & Simulation (HPC*S).

39. Eng, J. H., Saudi, A., Sulaiman, J. (2018). Numerical Evaluation of Quarter-Sweep SOR Iteration for Solving Poisson Image Blending Problem. *2018 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*.

40. (2003) The Finite Element Method for the Poisson Equation, *Texts in Applied Mathematics*, 44, pp. 46–91.

41. Ern, A., Guermond, J.L. (2004), Theory and practice of finite elements, *Springer*.

42. Lapaul, S., de Lustrac, A., Bouillault, F. (1996). Solving the Poisson's and Schrodinger's equations to calculate the electron states in quantum nanostructures using the finite element method, *IEEE Transactions on Magnetics*, 32(3), pp. 1018–1021.

43. Charles Crook (2013), Finite Element Methods for the Poisson Equation and its Applications [Електронний ресурс] – Ркжим доступу: http://educ.jmu.edu/~ohmx/REU/crookreport.pdf

44. Garcia-Loureiro, A.J., Pena, T.F., Lopez-Gonzalez, J.M., Prat, L. (2000) Parallel finite element method to solve the 3D Poisson equation and its application to abrupt heterojunction bipolar transistors, *Int. J. Numer. Meth. Engng*., 49, pp. 639-652.

45. Clarke, L., Glendinning, I., Hempel, R. (1994) The MPI Message Passing Interface Standard. In: Decker, K.M., Rehmann, R.M. (eds) *Programming Environments for Massively Parallel Distributed Systems*. Monte Verità. Birkhäuser, Basel.

46. Nayroles, B, Touzot, G, Villon, P (1992) Generalizing the finite element method, diffuse approximation and diffuse elements, *Comput Mech*, 10, pp. 307–318

47. Belytschko, T., Lu, Y. Y., Gu, L. (1994) Element-free Galerkin methods, *International Journal for Numerical Methods in Engineering*, 37(2), pp. 229–256.

48. Onate, E, Idelsohn, S, Zienkiewicz, OC, Taylo,r RL, Sacco, C (1996) A stabilized finite point method for analysis of fluid mechanics problems, *Comput Methods Appl Mech Eng*, 139(1/4), pp. 315–346

49. Löhner, R., Oñate, E. (1998). An advancing front point generation technique, *Communications in Numerical Methods in Engineering*. 14(12), pp. 1097–1108.

50. Koshizuka, S, Oka, Y (1996) Moving-particle semi-implicit method for fragmentation of incompressible fluid, *Nucl Sci Eng*, 123(3), pp. 421–434

51. Koshizuka, S., Ikeda, H., Oka, Y. (1999) Numerical analysis of fragmentation mechanisms in vapor explosions, *Nuclear Engineering and Design*, 189, pp. 423–433.

52. Zhang, S., Morita, K., Fukuda, K., Shirakawa, N. (2006) An improved MPS method for numerical simulations of convective heat transfer problems, *Int. J. Numer. Meth. Fluids*, 51, pp. 31-47.

53. Ataie-Ashtiani, B., Farhadi, L.(2006) A stable moving particle semi-implicit method for free surface flows, *Fluid Dynamics Research*, 38, pp. 241–256.

54. Shakibaeinia, A., Jin, Y.-C. (2010) A weakly compressible MPS method for modeling of open-boundary free-surface flow, *Int. J. Numer. Meth. Fluids*, 63, pp. 1208-1232.

55. Suzuki, Y., Koshizuka, S., Oka, Y. (2007) Hamiltonian moving-particle semi-implicit (HMPS) method for incompressible fluid flows, *Computer Methods in Applied Mechanics and Engineering*, 196(29–30), pp. 2876-2894.

56. Khayyer, A., Gotoh, H. (2008) Development of CMPS Method for Accurate Water-Surface Tracking in Breaking Waves, *Coastal Engineering Journal*, 50(2), pp. 179-207.

57. Jandaghian, M., Shakibaeinia, A. (2020) An enhanced weakly-compressible MPS method for free-surface flows, *Computer Methods in Applied Mechanics and Engineering*, 360, 112771.

58. Khayyer, A., Gotoh, H. (2009) Modified Moving Particle Semi-implicit methods for the prediction of 2D wave impact pressure, *Coastal Engineering*, 56(4), pp. 419-440.

59. Kondo, M., Koshizuka, S. (2011), Improvement of stability in moving particle semi-implicit method, *Int. J. Numer. Meth. Fluids*, 65, pp. 638-654.

60. Khayyer, A., Gotoh, H. (2010) A higher order Laplacian model for enhancement and stabilization of pressure calculation by the MPS method, *Applied Ocean Research*, 32(1), pp. 124-131.

61. Xu, T., Jin, Y.-C. (2019) Improvement of a projection-based particle method in free-surface flows by improved Laplacian model and stabilization techniques, *Computers & Fluids*, 191, 104235.

62. Nabian, M.A., Farhadi, L. (2016) Multiphase Mesh-Free Particle Method for Simulating Granular Flows and Sediment Transport, *Journal of Hydraulic Engineering*.

63. Xu, T. and Jin, Y.-C. (2021) "Two-dimensional continuum modelling granular column collapse by non-local peridynamics in a mesh-free method with μ(I) rheology," *Journal of Fluid Mechanics*. Cambridge University Press, 917, p. A51.

64. Xu, T., Li, S.S. (2022) Development of a non-local partial Peridynamic explicit mesh-free incompressible method and its validation for simulating dry dense granular flows. *Acta Geotech*.

65. Gingold, R.A., Monaghan, J.J. (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars, *Mon. Not. R. Astron. Soc*., 181(3), pp. 375–389.

66. Lucy, L.B. (1977) A numerical approach to the testing of the fission hypothesis, *Astron. J*., 82, pp. 1013–1024.

67. Toyoshi, T, Wada, Y, Kikuchi, M (2011) Solid-liquid flows simulation for debris avalanche analysis, *Key Eng Mater*, 462 & 463, pp. 855–860

68. Gotoh, H, Ikari, H, Yasuoka, T (2009) Simulation of armor blocks in front of caisson breakwater by DEM-MPS hybrid model. *Proceedings of 19th international offshore and polar engineering conference, Osaka*, pp 365–370

69. Canelas, R.B., Crespo, A.J.C., Domínguez, J.M., Ferreira, R.M.L., Gómez-Gesteira, M. (2016) SPH-DCDEM model for arbitrary geometries in free surface solid-fluid flows, *Comput Phys Commun*, 202, pp. 131–140

70. Hori, C, Gotoh, H, Ikari, H, Khayyer, A (2011) GPU-acceleration for moving particle semi-implicit method, *Comput Fluids*, 51(1), pp. 174–183.

71. Kakuda, K., Nagashima, T., Hayashi, Y., Obara, S., Toyotani, J., Miura, S., Matsuda, S. (2013) Three-dimensional fluid flow simulations using GPU-based particle method, *Comput Model Eng Sci*, 93(5), pp. 363–376

72. Qiu, L.C. (2014) OpenCL-based GPU acceleration of ISPH simulation for incompressible flows, *Appl Mech Mater*, 444, pp.380–384.

73. Ulam, S. (1950) Random processes and transformations, *Proceedings of the International Congress on Mathematics*, 2, pp. 264-275.

74. Ulam, S. (1962) On some mathematical problems connected with patterns of growth in figures, in *Mathematical Problems in the Biological Sciences*, pp. 215–224.

75. von Neumann, J (1966) T*he Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana.

76. Turing, A.M (1936) On computable numbers, with an application to the Entscheidungsproblem, *Proc. Lond. Math. Soc*., 42, pp. 230-265.

77. Zuse, K. (1970) *Calculating Space*, MIT Technical Translation. Translated by Aztec School of Languages, Inc. Cambridge, Massachusetts, USA.

78. Hardy, J., O. de Pazzis and Y. Pomeau (1976) Molcular dynamics of a lattice gas: Transport properties and time correlation functions, *Phys. Rev.*, A13, pp. 1949-1961.

79. Zuse, K. (1982) *The computing universe, Int. J. Th. Phys*., 21(6/7), pp. 589-600.

80. Gardner, M (1970) The fantastic combinations of John Conway's new solitaire game 'life', *Mathematical Games. Scientific American*. 223 (4), pp. 120—123.

81. Gardner, M. (1970) Geometric fallacies: hidden errors pave the road to absurd conclusions, *Sci. Am.*, 224(4), pp. 114-117.

82. Gardner, M. (1971) On cellular automata, self-reproduction, the Garden of Eden and the game “life", *Sci. Am.*, 224(2), pp. 112-117.

83. Gardner, M. (1971) The orders of infinity, the topological nature of dimensions and “supertasks", *Sci. Am.*, 224(3), pp. 106-109.

84. Broadwell, J.E. (1964) Study of Rarefied Shear Flow by the Discrete Velocity Method, *J. Fluid Mech.*, 19, pp. 401-414.

85. Hardy, J., Pomeau, Yv, de Pazzis, O. (1973) Time Evolution of a Two-Dimensional Classical Lattice System, *Phys. Rev. Lett.*, 31, pp. 276-279.

86. Hardy, J., Pomeau, Yv, de Pazzis, O. (1973) Time Evolution of a TwoDimensional Model System. 1. Invariant States and Time Correlation Functions, *J. Math. Phys*., 14, pp. 1746-1759

87. Succi S. (2001) *The Lattice Boltzmann Equation, for fluid dynamics and beyond*, Oxford Science Publications.

88. Frisch, U., Hasslacher, B., Pomeau, Y. (1986) Lattice-gas automata for the Navier–Stokes equations, *Phys. Rev. Lett.*, 56, pp. 1505–1508

89. Dieter, A., Wolf-Gladrow, D. (2000). *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*, Springer

90. McNamara, G., Zanetti, G. (1988) Use of a Boltzmann equation to simulate lattice-gas automata, *Phys. Rev. Lett.*, 61, 2332

91. Higuera, F., Jimenez, J. (1989) Boltzmann approach to lattice gas simulations, *Europhys. Lett.*, 9, pp. 663–668.

92. Bhatnagar, P. L., Gross, E. P. and Krook, M. (1954) A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems, *Phys. Rev. American Physical Society*, 94, pp. 511–525.

93. Koelman J.M.V.A. (1991) A simple lattice Boltzmann scheme for Navier–Stokes fluid flow, *Europhys. Lett.*, 15, pp. 603–607.

94. Chen, S., Chen, H., Martinez, Matthaeus (1991) Lattice Boltzmann model for simulation of magnetohydrodynamics, *Phys. Rev. Lett.*, 67, pp. 3776–3780.

95. Qian, Y.H., d'Humieres, D., Lallamand, P. (1992) Lattice BGK models for Navier–Stokes equation, *Europhys. Lett.*, 17, pp. 479–484.

96. Chen, H., Chen, S., Matthaeus, W.H. (1992) Recovery of the Navier–Stokes equation using a lattice-gas Boltzmann method, *Phys.Rev*., A 45, pp. 5339–5342

97. Wolf-Gladrow, D. (2005) *Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction*, Bremerhaven: Alfred Wegener Institute for Polar and Marine, p. 311.

98. Wolfram, S. (1986) *Cellular automaton fluids 1: Basic theory*, J. Stat. Phys., 45, p. 471.

99. Lavallee, P., Boon, J.P., Noullez, A. (1991) Boundaries in lattice gas flows, *Physica D*, 47, p. 233.

100. Ziegler, D.P. (1993) Boundary conditions for lattice Boltzmann simulations, *J. Stat. Phys.*, 71, pp. 1171–1177.

101. Cornubert, R., D'Humi`eresm, D., Levermore, D. (1991) A knudsen layer theory for lattice gases, *Physica D*, 47, pp. 241–259.

102. Ginzbourg, I., Adler, P.M. (1994) Boundary flow condition analysis for threedimensional lattice Boltzmann model, *J. Phys. II France*, 4, pp. 191–214.

103. Chen, S., Mart`ınez, D., Mei, R. (1996) On boundary conditions in lattice Boltzmann methods, *Phys. Fluids*, 8(9), pp. 2527–2536.

104. Inamuro, T., Yoshina, M., Ogino, F. (1995) A non-slip boundary condition for lattice Boltzmann simulations, *Phys. Fluids*, 7(12), pp. 2928–2930.

105. Noble, D.R., Georgiadis, J.G., Buckius, R.O. (1995) Direct assessment of lattice Boltzmann hydrodynamics and boundary conditions for recirculating flows, *J. Stat. Phys.*, 81, pp. 17–33.

106. Filippova, O., H¨anel, D. (1998) Boundary-fitting and local grid refinement for lattice-BGK models, *Int. J. Mod. Phys.*, 9, pp. 1271–1279.

107. Filippova, O., H¨anel, D. (1998) Grid refinement for lattice-bgk models. *J. Comp. Phys*., 147, pp. 219–228.

108. Mei, R., Luo, L.-S., Shyy, W. (1999) An accurate curved boundary treatment in the lattice Boltzmann method, *J. Comp. Phys.*, 155, pp. 307–330.

109. Mei, R., Shyy, W., Yu, D., Luo, L.-S. (2000) Lattice Boltzmann method for 3-d flows with curved boundary, *J. Comp. Phys*., 161, pp. 680–699..

110. Bouzidi. M., Firdaouss, M., Lallemand, P. (2001) Momentum transfer of a Boltzmann-lattice fluid with boundaries, *Phys. Fluids*, 13(11), pp. 3452–3459.

111. Ginzbourg, I., d'Humi`eres, D. (2003) The multireflection boundary conditions for lattice Boltzmann models, *Physical Review*, 68, 066614.

112. Succi S., Benzi, R., Higuera, F. (1991) The lattice Boltzmann equation: a new tool for computational fluiddynamics, *Physica D*, 47, pp. 219-230.

113. Yu, D., Mei, R. and Shyy, W. (2003) A Unified Boundary Treatment in Lattice Boltzmann Method, in *41st Aerospace Sciences Meeting and Exhibit*.

114. Skordos P., Initial and Boundary conditions for the lattice Boltzmann method, *Physical review E*., 48, pp. 4823-4842.

115. Martinez D. O., Matthaeus, W.H., Chen, S. (1994) Comparison of spectral method and lattice Boltzmann simulations of two-dimensional hydrodynamics, *Physics of Fluid*, 6, pp. 1285-1290.

116. Hou, S., Zou, Q., Chen, S., Doolen, G., Cogley, A.C. (1995) Simulation of cavity flow by the lattice Boltzmann method, *Journal of computational physics*, 118(2), pp. 329-347.

117. Shi, B., He, N., Wang, N., Guo, Z. (2003) Lattice Boltzmann Simulations of Fluid Flows, *Springer-Verlag, BerlinHeidelberg*, 24, pp. 322–332.

118. De, S., Nagendra, K., Lakshmisha, K.N. (2009) Simulation of laminar flow in a three-dimensional lid-driven cavity by lattice Boltzmann method, *Int. J. Numer. Methods Heat Fluid Flow*, 19, pp. 790–815.

119. Patil, D.V., Lakshmisha, K.N., (2009) Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh, *J. Comput. Phys*., 228, pp. 5262–5279.

120. He, X. (1997) Theory of the lattice Boltzmann method: from the Boltzmann equation to the lattice Boltzmann equation, *Phys. Rev.*, 56(6), pp. 6811-6817.

121. Zou, Q, He, X. (1997) On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, *Phys. of Fluids*, 9, pp. 1591-1598.

122. He, X., Luo, L. (1997) Lattice Boltzmann model for the incompressible Navier-Stokes equation, *Journal of statistical physics*, 88(3), pp. 927-944.

123. Breuer, M., Bernsdorf, J., Zeiser, T., Durst, F. (2000) Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finitevolume, *Int. J. Heat Fluid Flow*, 21, pp. 186–196.

124. Yoshino, M., Matsuda, Y., Shao, C. (2004) Comparison of accuracy and efficiency between the lattice Boltzmann method and the finite difference Method in viscous/thermal fluid flows, *Int. J. Comput. Fluid Dyn.*, 18, pp. 333–345

125. Geller, S., Krafczyk, M., Tolke, J., Turek, S., Hron, J. (2006) Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows, *Comput. Fluids*, 35, pp. 888–897

126. Martinez, D.O., Matthaeus, W.H., Chen, S. (1994) Comparison of spectral method and lattice Boltzmann simulations of twodimensional hydrodynamics, *Phys. Fluids*, 6 (1994), pp. 1285–1298

127. He, X., Doolen, G. (1997) Lattice Boltzmann method on curvilinear coordinates system: flow around a circular cylinder, *J. Comput. Phys.*, 134, pp. 306–315

128. Perumal, D.A., Kumar, G.V.S., Dass, A.K. (2014) Lattice Boltzmann simulation of flow over a circular cylinder at moderate Reynolds numbers, *Therm. Sci.*, 18, pp. 1235–1246.

129. Chen, S., Doolen, G. (1998) Lattice Boltzmann method for fluid flows, *Annu. Rev. Fluid Mech.*, 30, pp. 329-364.

130. Tolke, J., Krafczyk, M., Rank, E. (2002) A multigrid-solver for the discrete Boltzmann equation, *J. Stat. Phys.*, 117, pp. 573–591.

131. Mavriplis, D.J. (2006) Multigrid solution of the steady-state lattice Boltzmann equation, *Comput. Fluids*, 35, pp. 793–804.

132. He, X., Luo, L.S., Dembo, M. (1996) Some progress in lattice Boltzmann method. Part I. Non-uniform mesh grids, *J. Comput. Phys.*, 129, pp. 357–363

133. Kuznik, F., Vareilles, J., Rusaouen, G., Krauss, G. (2007) A doublepopulation lattice Boltzmann method with non-uniform mesh for the simulation of natural convection in a square cavity, *Int. J. Heat Fluid Flow*, 28, pp. 862–870.

134. Alexander, F.J., Chen, H., Chen, S., Doolen, G.D. (1992) Lattice Boltzmann model for compressible fluids, *Phys. Rev.*, A 46, pp. 1967–1970.

135. Chen, Y., Ohashi, H., Akiyama, M. (1994) Thermal lattice Bhatnagar–Gross–Krook model without nonlinear derivations in macrodynamic equations, *Phys. Rev. E*, 50, pp. 2776–2783.

136. Yan, G., Chen, Y., Hu, S. (1999) Simple lattice Boltzmann model for simulating flows with shock wave, *Phys. Rev. E*, 59, pp. 454–459.

137. Yu, H., Zhao, K. (2000) Lattice Boltzmann method for compressible flows with high Mach numbers, *Phys. Rev. E*, 61, pp. 3867–3870.

138. Palmer, B., Rector, D. (2000) Lattice Boltzmann algorithm for simulating thermal flow in compressible fluids, *J. Comput. Phys.*, 161, pp. 1–20.

139. Kataoka, T., Tsutahara, M. (2004) Lattice Boltzmann model for the compressible Navier–Stokes equations with flexible specificheat ratio, *Phys. Rev. E*, 69, pp. 0357011–0357014

140. He, B., Chen, Y., Feng, W., Li, Q., Song, A., Wang, Y., Zhang, M., Zhang, W. (2012) Compressible lattice Boltzmann method and applications, *Int. J. Numer. Anal. Model*, 9, pp. 410–418.

141. Li, K., Zhong, C. (2015) A lattice Boltzmann model for simulation of compressible flows, *Int. J. Numer. Meth. Fluids*, 77, pp. 334–357.

142. Li, W., Wei, X., & Kaufman, A. (2003) Implementing lattice boltzmann computation on graphics hardware, *Visual Computer*, 19, pp. 444–456.

143. Mawson, M. J., Revell, A. J. (2014) Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs, *Computer Physics Communications*, 185(10), pp. 2566 –2574.

144. Habich, J., Zeiser, T., Hager, G., Wellein, G. (2011) Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA, *Advances in Engineering Software*, 42(5), pp. 266 – 272.

145. Herschlag, G., Lee, S., S. Vetter, J., & Randles, A. (2018) GPU Data Access on Complex Geometries for D3Q19 Lattice Boltzmann Method. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 825–834.

146. CUDA [Електронний ресурс]. – Режим доступу: https://developer.nvidia.com/cuda-zone

147. Tran, N.-P., Lee, M. and Hong, S. (2017) Performance Optimization of 3D Lattice Boltzmann Flow Solver on a GPU, *Scientific Programming*. Hindawi, 2017, p. 1205892

148. Mendoza, M., Munoz, J. (2010) Three-dimensional lattice Boltzmann model for electrodynamics, *Physical Review E. American Physical Society*, 82(5), p. 056708.

149. Hauser, A.,Verhey, J. (2019) Comparison of the lattice-Boltzmann model with the finite-difference timedomain method for electrodynamics, *Physical Review E*, 99(3), p. 033301.

150. Succi, S., Vergassola, M., Benzi, R. (1991) Lattice Boltzmann scheme for two-dimensional magnetohydrodynamics, *Physical Review A*., 43(8), pp. 4521-4524.

151. Chen, S., Chen, H., Martnez, D., Matthaeus, W. (1991) Lattice Boltzmann model for simulation of magnetohydrodynamics, *Physical Review Letters*, 67(27), pp. 3776-3779.

152. Mendoza, M., Boghosian, B., Herrmann, H. J., Succi, S. (2010) Fast lattice Boltzmann solver for relativistic hydrodynamics, Physical review letters, 105(1), p. 014502.

153. Ma, Y., Dong, S., Tan, H. (2011) Lattice Boltzmann method for one-dimensional radiation transfer, *Physical Review E*, 84(1), p. 016704.

154. De Rosis, A. (2020) Modeling epidemics by the lattice Boltzmann method, *Physical Review E.*, 102(2), p. 023301.

155. Xue, Y., Liu, P., Tao, Y., Tang, X. (2017) Abnormal prediction of dense crowd videos by a purpose-driven lattice Boltzmann model, *International Journal of Applied Mathematics and Computer Science*, 27(1), pp. 181–194.

156. Mehta, U., Kutler, P. (07 1984) Computational aerodynamics and artificial intelligence.

157. Hopfield, J. (05 1982) Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proceedings of the National Academy of Sciences of the United States of America*, 79, pp. 2554–2558.

158. Hinton, G. and Sejnowski, T. (01 1986) Learning and relearning in Boltzmann machines, *Parallel Distributed Processing*, 1.

159. Gardner, E. (1988) The space of interactions in neural network models, *J. Phys. A.*, 21, pp. 257-270.

160. Teo, C., Lim, K., Hong, G., Yeo, M. (1991) A neural net approach in analysing photographs in PIV. *IEEE Sys. Man. Cybern.*, 3, pp. 1535–1538

161. Grant, I., Pan, X. (1995) An investigation of the performance of multi layer, neural networks applied to the analysis of PIV images, *Exp. Fluids*, 19, pp. 159–166

162. Dissanayake, M. W. M. G., Phan-Thien, N. (1994) Neural-network-based approximations for solving partial differential equations, *Communications in Numerical Methods in Engineering*, 10(3), pp. 195-201.

163. Sirignano, J., Spiliopoulos, K. (2018) DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, 375, pp.1339–1364.

164. Lee, H., Kang, I.S. (1990) Neural algorithm for solving differential equations, *Journal of Computational Physics*, 91(1), pp. 110-131.

165. Smaoui, N., Al-Enezi, S. (2004) Modelling the dynamics of nonlinear partial differential equations using neural networks, *Journal of Computational and Applied Mathematics*, 170(1), pp. 27-58.

166. Bar-Sinai, Y., Hoyer, S., Hickey, J., Brenner, M.P. (2019) Learning data-driven discretizations for partial differential equations, *Proceedings of the National Academy of Sciences*, 116(31), pp. 15344–15349.

167. Stevens, B., Colonius, T. (2020) Enhancement of shock-capturing methods via machine learning, *Theoretical and Computational Fluid Dynamics*, 34, pp. 483–496.

168. Jeon, J., Kim, S.J. (2021) FVM Network to reduce computational cost of CFD simulation, *arXiv preprint arXiv:2105.03332.*

169. Stevens, B., Colonius, T. (2020) Finitenet: A fully convolutional LSTM network architecture for time-dependent partial differential equations, *arXiv preprint arXiv:2002.03014.*

170. Kochkov, D., Smith. J.A., Alieva, A., Wang, Q., Brenner, M.P., Hoyer, S. (2021) Machine learning-accelerated computational fluid dynamics, *Proceedings of the National Academy of Sciences*, 118, p. e2101784118.

171. Chandler, G.J., Kerswell, R.R. (2013) Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow, *Journal of Fluid Mechanics*, 722, pp. 554–595.

172. Shan, T., Tang, W., Dang, X., Li, M., Yang, F., Xu, S., Wu, J. (2017) Study on a Poisson's equation solver based on deep learning technique, *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pp. 1–3.

173. Zhang, Z., Zhang, L., Sun, Z., Erickson, N., From, R., Fan, J. (2019) Solving Poisson's equation using deep learning in particle simulation of PN junction, *2019 Joint International Symposium on Electromagnetic Compatibility, Sapporo and Asia-Pacific International Symposium on Electromagnetic Compatibility (EMC Sapporo/APEMC)*,pp. 305–308.

174. Ajuria, E., Alguacil, A., Bauerheim, M., Misdariis, A., Cuenot, B., Benazera, E. (2020) Towards a hybrid computational strategy based on deep learning for incompressible flows, *AIAA AVIATION Forum, June 15–19*, pp. 1–17.

175. Weymouth, G.D. (2022) Data-driven multi-grid solver for accelerated pressure projection, *Computers & Fluids*, 246, p. 105620.

176. Ozbay, A., Hamzehloo, A., Laizet, S., Tzirakis, P., Rizos, G., Schuller, B. (2021) Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh, *Data-Centric Engineering*, 2, p. E6.

177. Craft, T., Launder, B., Suga, K. (1996) Development and application of a cubic eddy-viscosity model of turbulence, *International Journal of Heat and Fluid Flow*, 17 (2), pp. 108-115.

178. Dow, E., Wang, Q. (2011) Quantification of structural uncertainties in the k-w turbulence model, In *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.

179. Parish, E. J., Duraisamy, K. (2016) A paradigm for data-driven predictive modeling using field inversion and machine learning, *Journal of Computational Physics*, 305, pp. 758-774.

180. Xiao, H., Wu, J.-L., Wang, J.-X., Sun, R., Roy, C. (2016) Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed bayesian approach, *Journal of Computational Physics*, 324, pp. 115-136.

181. Wu, J.-L., Wang, J.-X., Xiao, H. (2015) A Bayesian calibration-prediction method for reducing model-form uncertainties with application in RANS simulations. *Flow, Turbulence and Combustion*, 97.

182. Wang, J.-X., Wu, J.-L. and Xiao, H. (2017) Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, *Phys. Rev. Fluids. American Physical Society*, 2, p. 034603.

183. Duraisamy, K., Zhang, Z. J., Singh, A. P. (2015) New approaches in turbulence and transition modeling using data-driven techniques, in *53rd AIAA Aerospace Sciences Meeting*.

184. Guo, X., Li, W.,Iorio F. (2016) Convolutional neural networks for steady flow approximation, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA: Association for Computing Machinery)* pp 481-490.

185. Bedrunka, M. C., Wilde, D., Kliemank, M., Reith, D., Foysi, H., Kramer, A. (2021) Lettuce: Pytorch-based lattice boltzmann framework, in Jagode, H. et al. (eds) *High Performance Computing*. Cham: Springer International Publishing, pp. 40–55.

186. Huang, K. (1987) *Statistical Mechanics, 2 ed.*, John Wiley & Sons.

187. Полак, Л. С. (1987) *Людвиг Больцман*, М.: Наука.

188. Dunweg, B., Ladd, A. (2008) Lattice Boltzmann simulation of soft matter system, *Advanced Computer Simulation Approaches for Soft Matter Sciences*, 221, pp. 86-166.

189. Shan, X., Yuan, X.-F., Chen, H. (2006) Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation, *Journal of Fluid Mechanics*, 550, pp. 413-441.

190. Chapman, S., Cowling, T.G. (1970) *The Mathematical Theory of Non-Uniform Gases (3rd ed.)*, Cambridge University Press.

191. Batchelor, G. K. (1967), *An Introduction to Fluid Dynamics*, Cambridge University Press.

192. Sterling, J.D., Chen, S. (1996) Stability analysis of lattice Boltzmann methods, *J. Computational Physics*, 123, 196-206.

193. Mussa, M., Abdullah, S., Azwadi, C.S., Muhamad, N., Sopian, K. (2008) Numerical simulation of lid-driven cavity flow using the lattice Boltzmann method, *Applied Mathematics*, 13, pp. 236-240.

194. Wolf-Gladrow, D. (2005) *Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction*, Bremerhaven: Alfred Wegener Institute for Polar and Marine, pp. 163-169.

195. He, X., Zou, Q., Luo, L.-S., Dembo, M. (1997) Analytic solutions of simple flows and analysis of nonslip boundary conditions for the Lattice Boltzmann BGK model, *J. Stat. Phys*, 87, pp. 115-136.

196. Liu, H., Zhou, J.G., Burrows, R. (2012). Inlet and outlet boundary conditions for the Lattice-Boltzmann modelling of shallow water flows, *Progress in Computational Fluid Dynamics*, 12, pp. 11-18.

197. Geier, M., Schönherr, M., Pasquali, A., Krafczyk, M. (2015) The cumulant lattice Boltzmann equation in three dimensions: Theory and validation, *Computers & Mathematics with Applications*, 70(4), pp. 507–547.

198. Horwitz, J.A., Vanka, S.P., Kumar, P. (2019) LBM simulations of dispersed multiphase flows in a channel: role of a pressure Poisson equation, *Proceedings of the ASME-JSME-KMSE, 2019*.

199. Ansumali, S., Karlin, I. V., Öttinger, H. C. (2003). Minimal entropic kinetic models for hydrodynamics, *Europhysics Letters (EPL)*, 63(6), pp. 798–804.

200. Hosseini, S.A., Karlin, I. (2023). Entropic equilibrium for the lattice Boltzmann method: Hydrodynamics and numerical properties. *arXiv preprint arXiv.2303.08163.*

201. Karlin, I.V., Gorban, A.N., Succi, S., Boffi, V. (1998) Maximum Entropy Principle for Lattice Kinetic Equations, *Physical Review Letters*, 81(6).

202. Schofield, S.P., Christon, M. A., Dyadechko, V. (2010) Multi-material incompressible flow simulation using the moment-of-fluid-method, *International Journal for Numerical Methods in Fluids*, 63, pp. 931-952.

203. Yang, Z.R., Yang, Z. (2014) *Comprehensive Biomedical Physics*, Karolinska Institute, Stockholm, Sweden.

204. Tai, Y. (2021). A Survey Of Regression Algorithms And Connections With Deep Learning. *arXiv preprint arXiv.2104.12647*.

205. Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 37, pp. 448–456.

206. Cybenko, G. V. (1989) Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, 2, pp. 303–314.

207. Lin, T.-Y. et al. (2017) Feature Pyramid Networks for Object Detection, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944.

208. TensorFlow [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org/.

209. Duchi, J., Hazan. E., Singer, Y. (2011) Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, 12, pp. 2121-2159.

210. Bishop, C.M. (2006) *Pattern Recognition and Machine Learning*, Berlin, Heidelberg: Springer-Verlag.

211. Pedregosa F. (2011) Scikit-learn: machine learning in Python, *Journal of Machine Learning Research*, 12, pp. 2825-2830.

212. Hulsemann, F., Kowarschik, F., Mohr, M., Rude, U. (2005) Parallel geometric multigrid, In A. M. Bruaset, P. Bjørstad, and A. Tveito (eds.), *Numerical*

Solution of Partial Differential Equations on Parallel Computers, volume X of Lecture Notes in Computational Science and Engineering. Springer-Verlag.

213. cuDNN [Електронний ресурс]. – Режим доступу: https://developer.nvidia.com/cudnn

214. TPU [Електронний ресурс]. – https://cloud.google.com/tpu/docs/intro-to-tpu

215. Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit [Електронний ресурс] – Режим доступу: https://www.xilinx.com/products/boards-and-kits/zcu104.html#specifications

216. AMD [Електронний ресурс] – Режим доступу: https://www.amd.com/en.html

217. Vitis AI [Електрониий ресурс] – Режим доступу: https://xilinx.github.io/Vitis-AI/index.html

218. PyQt6 [Електронний ресурс] – Режим доступу: https://www.riverbankcomputing.com/static/Docs/PyQt6/

219. Harris, C.R., Millman, K.J., van der Walt, S.J. (2020) Array programming with NumPy, *Nature*, 585, pp. 357–362.

220. Hunter, J.D. (2007) Matplotlib: A 2D Graphics Environment, *Computing in Science & Engineering*, 9(3), pp. 90-95.

221. A Simple File Format for NumPy Arrays [Електронний ресурс] – Режим доступу: https://github.com/numpy/numpy/blob/067cb067cb17a20422e51da908920a4fbb3ab851/doc/neps/nep-0001-npy-format.rst

222. Common Format and MIME Type for Comma-Separated Values (CSV) Files [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc4180

223. The JavaScript Object Notation (JSON) Data Interchange Format [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7159

224. Неттер Ф. (переклад Цегельского А.А.) (2004). *Атлас анатомії людини*, Львів: Наутілус. с. 592.

225. Vittoz, G., Oger, M. (2019) Comparisons of weakly-compressible and truly incompressible approaches for viscous flow into a high-order Cartesian-grid finite volume framework, *Journal of Computational Physics*, 1, pp. 100015-100048.

226. Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021) Physics-informed machine learning, *Nature Reviews Physics*.

227. Jin, X., Cai, S., Li, H., Karniadakis, G.E. (2021) NSFnets (NavierStokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, *Journal of Computational Physics*, 426, 109951.

228. Yin, M., Zheng, X., Humphrey, J.D., Karniadakis, G.E. (2021) Noninvasive inference of thrombus material properties with physicsinformed neural networks, *Computer Methods in Applied Mechanics and Engineering*, 375, 113603.

229. GitHub - NVIDIA/modulus: A PyTorch based deep-learning toolkit for developing DL models for physical systems [Електронний ресурс] – Режим доступу: https://github.com/NVIDIA/modulus

230. Rout, S., Dwivedi, V., Srinivasan, B. (2021). "Numerical Approximation in CFD Problems Using Physics Informed Machine Learning", *arXiv preprint arXiv:2111.02987*

231. Eric Aislan Antonelo; Camponogara, Eduardo; Laio Oriel Seman; Eduardo Rehbein de Souza; Jordanou, Jean P.; Hubner, Jomi F. (2021). "Physics-Informed Neural Nets for Control of Dynamical Systems", *arXiv preprint arXiv:2104.02556*.

232. Jaffe, Arthur M. (2006). The Millennium Grand Challenge in Mathematics, *Notices of the American Mathematical Society*, 53 (6), pp. 652–660.

233. Sutera, Salvatore P., Skalak, R. (1993). The History of Poiseuille's Law, *Annual Review of Fluid Mechanics*, 25, pp. 1–19.

234. OpenFOAM [Електронний ресурс] – Режим доступу: https://www.openfoam.com/

235. Ansys Fluent Fluid Simulation Software [Електронний ресурс] – Режим доступу: https://www.ansys.com/products/fluids/ansys-fluent

236. SimScale [Електронний ресурс] – Режим доступу: https://www.simscale.com/

237. COMSOL – Spftware for Multiphysics Simulation [Електронний ресурс] – Режим доступу: https://www.comsol.com/

ДОДАТОК А

Частина програмного коду

Лістинг А.1 – Модуль main.py

```
import sys

from PyQt6 import QtCore, QtGui, QtWidgets

import main_window

class ExampleApp(QtWidgets.QMainWindow, window.Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)



def main():
    app = QtWidgets.QApplication(sys.argv)
    window = ExampleApp()
    window.show()
    app.exec()
```

```
if __name__ == '__main__':
    main()
```

Лістинг А.2 – Модуль main_window.py

```
from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(962, 809)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayoutWidget = QtWidgets.QWidget(parent=self.centralwidget)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(0, 0, 258, 572))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)

self.verticalLayout.setSizeConstraint(QtWidgets.QLayout.SizeConstraint.SetNoConstra
int)
        self.verticalLayout.setContentsMargins(10, 10, 10, 10)
```

```
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_7 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label_7.setObjectName("label_7")
        self.verticalLayout.addWidget(self.label_7)
        self.doubleSpinBox_4 =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)

self.doubleSpinBox_4.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRight)
        self.doubleSpinBox_4.setDecimals(2)
        self.doubleSpinBox_4.setMaximum(999999.99)
        self.doubleSpinBox_4.setProperty("value", 1000.0)
        self.doubleSpinBox_4.setObjectName("doubleSpinBox_4")
        self.verticalLayout.addWidget(self.doubleSpinBox_4)
        self.label = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label.setMinimumSize(QtCore.QSize(0, 0))
        self.label.setMaximumSize(QtCore.QSize(16777215, 35))

self.label.setAlignment(QtCore.Qt.AlignmentFlag.AlignLeading|QtCore.Qt.AlignmentF
lag.AlignLeft|QtCore.Qt.AlignmentFlag.AlignVCenter)
        self.label.setObjectName("label")
        self.verticalLayout.addWidget(self.label, 0, QtCore.Qt.AlignmentFlag.AlignTop)
        self.doubleSpinBox =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)
        self.doubleSpinBox.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRight)
```

```
        self.doubleSpinBox.setDecimals(6)
        self.doubleSpinBox.setMaximum(1.0)
        self.doubleSpinBox.setProperty("value", 0.5)
        self.doubleSpinBox.setObjectName("doubleSpinBox")
        self.verticalLayout.addWidget(self.doubleSpinBox, 0,
QtCore.Qt.AlignmentFlag.AlignTop)
        self.label_2 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label_2.setObjectName("label_2")
        self.verticalLayout.addWidget(self.label_2)
        self.doubleSpinBox_2 =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)

self.doubleSpinBox_2.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRight)
        self.doubleSpinBox_2.setDecimals(2)
        self.doubleSpinBox_2.setMaximum(9999999999.99)
        self.doubleSpinBox_2.setProperty("value", 1000.0)
        self.doubleSpinBox_2.setObjectName("doubleSpinBox_2")
        self.verticalLayout.addWidget(self.doubleSpinBox_2)
        self.label_3 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label_3.setObjectName("label_3")
        self.verticalLayout.addWidget(self.label_3)
        self.doubleSpinBox_3 =
QtWidgets.QDoubleSpinBox(parent=self.verticalLayoutWidget)
```

```
self.doubleSpinBox_3.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRight)
        self.doubleSpinBox_3.setDecimals(6)
        self.doubleSpinBox_3.setObjectName("doubleSpinBox_3")
        self.verticalLayout.addWidget(self.doubleSpinBox_3)
        self.checkBox = QtWidgets.QCheckBox(parent=self.verticalLayoutWidget)
        self.checkBox.setObjectName("checkBox")
        self.verticalLayout.addWidget(self.checkBox)
        self.checkBox_2 = QtWidgets.QCheckBox(parent=self.verticalLayoutWidget)
        self.checkBox_2.setObjectName("checkBox_2")
        self.verticalLayout.addWidget(self.checkBox_2)
        self.label_4 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label_4.setObjectName("label_4")
        self.verticalLayout.addWidget(self.label_4)
        self.radioButton_2 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_2.setChecked(True)
        self.radioButton_2.setObjectName("radioButton_2")
        self.buttonGroup = QtWidgets.QButtonGroup(MainWindow)
        self.buttonGroup.setObjectName("buttonGroup")
        self.buttonGroup.addButton(self.radioButton_2)
        self.verticalLayout.addWidget(self.radioButton_2)
        self.radioButton = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton.setObjectName("radioButton")
```

```
self.buttonGroup.addButton(self.radioButton)
self.verticalLayout.addWidget(self.radioButton)
self.label_5 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_5.setObjectName("label_5")
self.verticalLayout.addWidget(self.label_5)
self.spinBox = QtWidgets.QSpinBox(parent=self.verticalLayoutWidget)
self.spinBox.setMaximum(99999999)
self.spinBox.setProperty("value", 1000)
self.spinBox.setObjectName("spinBox")
self.verticalLayout.addWidget(self.spinBox)
self.label_8 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_8.setObjectName("label_8")
self.verticalLayout.addWidget(self.label_8)
self.radioButton_3 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
self.radioButton_3.setChecked(True)
self.radioButton_3.setObjectName("radioButton_3")
self.buttonGroup_2 = QtWidgets.QButtonGroup(MainWindow)
self.buttonGroup_2.setObjectName("buttonGroup_2")
self.buttonGroup_2.addButton(self.radioButton_3)
self.verticalLayout.addWidget(self.radioButton_3)
self.radioButton_4 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
self.radioButton_4.setObjectName("radioButton_4")
self.buttonGroup_2.addButton(self.radioButton_4)
```

```
        self.verticalLayout.addWidget(self.radioButton_4)
        self.radioButton_5 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_5.setObjectName("radioButton_5")
        self.buttonGroup_2.addButton(self.radioButton_5)
        self.verticalLayout.addWidget(self.radioButton_5)
        self.radioButton_6 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_6.setObjectName("radioButton_6")
        self.buttonGroup_2.addButton(self.radioButton_6)
        self.verticalLayout.addWidget(self.radioButton_6)
        self.label_9 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
        self.label_9.setObjectName("label_9")
        self.verticalLayout.addWidget(self.label_9)
        self.radioButton_7 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_7.setObjectName("radioButton_7")
        self.verticalLayout.addWidget(self.radioButton_7)
        self.radioButton_8 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_8.setChecked(True)
        self.radioButton_8.setObjectName("radioButton_8")
        self.verticalLayout.addWidget(self.radioButton_8)
        self.radioButton_10                                                    =
QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_10.setObjectName("radioButton_10")
        self.verticalLayout.addWidget(self.radioButton_10)
```

```
        self.radioButton_9 = QtWidgets.QRadioButton(parent=self.verticalLayoutWidget)
        self.radioButton_9.setObjectName("radioButton_9")
        self.verticalLayout.addWidget(self.radioButton_9)
        self.label_6 = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(260, 0, 311, 381))
        self.label_6.setText("")
        self.label_6.setObjectName("label_6")
        self.verticalLayoutWidget_2 = QtWidgets.QWidget(parent=self.centralwidget)
        self.verticalLayoutWidget_2.setGeometry(QtCore.QRect(740, 0, 218, 115))
        self.verticalLayoutWidget_2.setObjectName("verticalLayoutWidget_2")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.verticalLayoutWidget_2)
        self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.pushButton = QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Expanding,
QtWidgets.QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
        self.pushButton.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setBold(False)
```

```
        font.setWeight(50)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        self.verticalLayout_2.addWidget(self.pushButton)
        self.pushButton_2 = QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum,
QtWidgets.QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_2.sizePolicy().hasHeightForWidth())
        self.pushButton_2.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(10)
        font.setBold(False)
        font.setUnderline(False)
        font.setWeight(50)
        font.setKerning(True)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setObjectName("pushButton_2")
        self.verticalLayout_2.addWidget(self.pushButton_2)
        self.pushButton_3 = QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
```

```
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Expanding, QtWidgets.QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_3.sizePolicy().hasHeightForWidth())
        self.pushButton_3.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(10)
        font.setBold(False)
        font.setUnderline(False)
        font.setWeight(50)
        font.setKerning(True)
        self.pushButton_3.setFont(font)
        self.pushButton_3.setObjectName("pushButton_3")
        self.verticalLayout_2.addWidget(self.pushButton_3)
        self.pushButton_4 = QtWidgets.QPushButton(parent=self.verticalLayoutWidget_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum, QtWidgets.QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_4.sizePolicy().hasHeightForWidth())
```

```
self.pushButton_4.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setFamily("MS Shell Dlg 2")
font.setPointSize(10)
font.setBold(False)
font.setUnderline(False)
font.setWeight(50)
font.setKerning(True)
self.pushButton_4.setFont(font)
self.pushButton_4.setObjectName("pushButton_4")
self.verticalLayout_2.addWidget(self.pushButton_4)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(parent=MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 962, 21))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Modifed LBM Modelling"))
        self.label_7.setText(_translate("MainWindow", "Початкова густина"))
        self.label.setText(_translate("MainWindow", "Параметр релаксації"))
        self.label_2.setText(_translate("MainWindow", "Число Рейнольдса"))
        self.label_3.setText(_translate("MainWindow", "Швидкість притоку"))
        self.checkBox.setText(_translate("MainWindow", "Використовувати корекцію швидкості"))
        self.checkBox_2.setText(_translate("MainWindow", "Використовувати нейронну мережу"))
        self.label_4.setText(_translate("MainWindow", "Апаратний прискорювач для нейронної мережі"))
        self.radioButton_2.setText(_translate("MainWindow", "GPU"))
        self.radioButton.setText(_translate("MainWindow", "NPU"))
        self.label_5.setText(_translate("MainWindow", "Кількість ітерацій моделювання"))
        self.label_8.setText(_translate("MainWindow", "Напрям притоку"))
        self.radioButton_3.setText(_translate("MainWindow", "Зверху"))
        self.radioButton_4.setText(_translate("MainWindow", "Внизу"))
        self.radioButton_5.setText(_translate("MainWindow", "Зліва"))
        self.radioButton_6.setText(_translate("MainWindow", "Справа"))
        self.label_9.setText(_translate("MainWindow", "Напрям витоку"))
```

```
        self.radioButton_7.setText(_translate("MainWindow", "Зверху"))
        self.radioButton_8.setText(_translate("MainWindow", "Внизу"))
        self.radioButton_10.setText(_translate("MainWindow", "Зліва"))
        self.radioButton_9.setText(_translate("MainWindow", "Справа"))
        self.pushButton.setText(_translate("MainWindow", "Завантажити обчислювальну область"))
        self.pushButton_2.setText(_translate("MainWindow", "Почати моделювання"))
        self.pushButton_3.setText(_translate("MainWindow", "Зупинити моделювання"))
        self.pushButton_4.setText(_translate("MainWindow", "Зберегти результати моделювання"))


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1309, 896)
        MainWindow.setDocumentMode(False)
        MainWindow.setDockNestingEnabled(False)
        MainWindow.setUnifiedTitleAndToolBarOnMac(False)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalSlider = QtWidgets.QSlider(parent=self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(160, 430, 641, 22))
        self.horizontalSlider.setMaximum(1000)
```

```
        self.horizontalSlider.setProperty("value", 300)
        self.horizontalSlider.setTracking(True)
        self.horizontalSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)
        self.horizontalSlider.setInvertedControls(False)

self.horizontalSlider.setTickPosition(QtWidgets.QSlider.TickPosition.TicksAbove)
        self.horizontalSlider.setTickInterval(20)
        self.horizontalSlider.setObjectName("horizontalSlider")
        self.label = QtWidgets.QLabel(parent=self.centralwidget)
        self.label.setGeometry(QtCore.QRect(140, 410, 51, 16))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(760, 410, 71, 16))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_2.setFont(font)
        self.label_2.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.verticalLayoutWidget = QtWidgets.QWidget(parent=self.centralwidget)
```

```
self.verticalLayoutWidget.setGeometry(QtCore.QRect(0, 0, 151, 361))
self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
self.verticalLayout.setContentsMargins(0, 0, 0, 0)
self.verticalLayout.setObjectName("verticalLayout")
self.label_3 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_3.setObjectName("label_3")
self.verticalLayout.addWidget(self.label_3)
self.label_4 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_4.setObjectName("label_4")
self.verticalLayout.addWidget(self.label_4)
self.label_6 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_6.setObjectName("label_6")
self.verticalLayout.addWidget(self.label_6)
self.label_8 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_8.setObjectName("label_8")
self.verticalLayout.addWidget(self.label_8)
self.label_9 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_9.setObjectName("label_9")
self.verticalLayout.addWidget(self.label_9)
self.label_7 = QtWidgets.QLabel(parent=self.verticalLayoutWidget)
self.label_7.setObjectName("label_7")
self.verticalLayout.addWidget(self.label_7)
```

```
        self.label_10 = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_10.setGeometry(QtCore.QRect(160, 40, 1131, 311))
        self.label_10.setText("")

self.label_10.setPixmap(QtGui.QPixmap("../../../PhD/final_disser/img/ui_viz/sum_vals-2.png"))
        self.label_10.setScaledContents(True)
        self.label_10.setObjectName("label_10")
        self.horizontalLayoutWidget = QtWidgets.QWidget(parent=self.centralwidget)
        self.horizontalLayoutWidget.setGeometry(QtCore.QRect(160, 0, 1121, 41))
        self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.label_11 = QtWidgets.QLabel(parent=self.horizontalLayoutWidget)
        self.label_11.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label_11.setObjectName("label_11")
        self.horizontalLayout.addWidget(self.label_11)
        self.label_12 = QtWidgets.QLabel(parent=self.horizontalLayoutWidget)
        self.label_12.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label_12.setObjectName("label_12")
        self.horizontalLayout.addWidget(self.label_12)
        self.label_13 = QtWidgets.QLabel(parent=self.horizontalLayoutWidget)
```

```
        self.label_13.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label_13.setObjectName("label_13")
        self.horizontalLayout.addWidget(self.label_13)
        self.label_14 = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_14.setGeometry(QtCore.QRect(160, 380, 211, 16))
        self.label_14.setObjectName("label_14")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(parent=MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1309, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Modified LBM
Modelling - Результати моделювання"))
        self.label.setText(_translate("MainWindow", "0"))
```

```
        self.label_2.setText(_translate("MainWindow", "1000"))
        self.label_3.setText(_translate("MainWindow", "Початкова густина: 1000.0"))
        self.label_4.setText(_translate("MainWindow", "Параметр релаксації: 0.5"))
        self.label_6.setText(_translate("MainWindow", "Швидкість притоку: 0.005"))
        self.label_8.setText(_translate("MainWindow", "Напрям притоку: справа"))
        self.label_9.setText(_translate("MainWindow", "Напрям витоку: зверху"))
        self.label_7.setText(_translate("MainWindow", "Апаратний прискорювач \n"
"для нейронної мережі: GPU"))
        self.label_11.setText(_translate("MainWindow", "Густина"))
        self.label_12.setText(_translate("MainWindow", "Швидкість"))
        self.label_13.setText(_translate("MainWindow", "Тиск"))
```

Лістинг А.3 – модуль datagen.py

```
import sys
sys.path.append('../')

from pathlib import Path
from itertools import product
from random import choice, shuffle

import numpy as np
from PIL import Image
```

```
import joblib


from sklearn.preprocessing import QuantileTransformer, MinMaxScaler

from tqdm.autonotebook import tqdm


from src.lbm_datagen import iterate_lbm_modelling, corrected_velocity_with_rho, make_ground_truth_with_rho

from src.geometry_dataset import distance_array


train_geos_folder = Path(r'G:\Projects\PhD\poisson-2d\data\geometries_2')

test_geos_folder = Path(r'G:\Projects\PhD\poisson-2d\data\test_geometries')


dataset_folder = Path('../data/96-rho-lbm-2')

dataset_folder.mkdir(exist_ok=True)

train_images = {p.name.split('.')[0]: Image.open(p) for p in train_geos_folder.glob('*.png')}



# # Train dataset


Res = [500, 1000, 2000]

v0s = [0.002, 0.004, 0.01]
```

```
num_iters = 600

return_step = 2


rhoo = 1


total_samples = len(train_images) * len(Res) * len(v0s) * (num_iters // return_step + 1)

total_samples


x_train = []

rho_arr_train = []

y_train = []

dist_arr_train = []


pbar = tqdm(total = total_samples)


for geom_num, geo in train_images.items():

    print(geom_num)

    geo_arr = np.array(geo.convert('1')).astype(np.uint8)

    dist_arr = distance_array(geo_arr)

    obs = geo_arr == 1

    for Re, v0 in product(Res, v0s):

        for step, b_part, rho, pressure_field in iterate_lbm_modelling(

                geo_arr, v0, rhoo, num_iters, Re, return_step=return_step):
```

```
            b_part[obs] = 0
            rho[obs] = 0
            pressure_field[obs] = 0

            # keep values in [-1, 1] range
            rho_k = max(np.abs(rho).max(), 1)
            rho = rho / rho_k

            b_k = max(np.abs(b_part).max(), 1)
            b_part = b_part / b_k

            pressure_field = pressure_field / (rho_k * b_k)

            x_train.append(b_part)
            rho_arr_train.append(rho)
            y_train.append(pressure_field)
            dist_arr_train.append(dist_arr)

            pbar.update()

t = list(zip(x_train, y_train, dist_arr_train, rho_arr_train))
shuffle(t)
x_train, y_train, dist_arr_train, rho_arr_train = zip(*t)
```

```
x_train = np.array(x_train)
y_train = np.array(y_train)
dist_arr_train = np.array(dist_arr_train)
rho_arr_train = np.array(rho_arr_train)
x_train.min(), x_train.max()
qt = QuantileTransformer(n_quantiles=500, output_distribution='normal')
min_max = MinMaxScaler((-1, 1))
u_tr1 = qt.fit_transform(y_train.reshape(-1, 1))
min_max.fit(u_tr1)
y = qt.transform(y_train.reshape(-1, 1))
y_train_scaled = min_max.transform(y).reshape(-1, 96, 96)
joblib.dump(qt, str(dataset_folder / 'qt.pkl'))
joblib.dump(min_max, str(dataset_folder / 'min_max.pkl'))
with open(dataset_folder / 'x_train.npy', 'wb') as f:
    np.save(f, x_train)
with open(dataset_folder / 'y_train.npy', 'wb') as f:
    np.save(f, y_train_scaled)
with open(dataset_folder / 'y_train_raw.npy', 'wb') as f:
    np.save(f, y_train)
with open(dataset_folder / 'dist_arr_train.npy', 'wb') as f:
    np.save(f, dist_arr_train)
with open(dataset_folder / 'rho_arr_train.npy', 'wb') as f:
```

```
    np.save(f, rho_arr_train)
```

Лістинг А.4 – модуль lbm_datagen.py

```
import numpy as np

from scipy.sparse import csr_matrix
import pyamg


col1 = np.array([0, 1, 2])
col2 = np.array([3, 4, 5])
col3 = np.array([6, 7, 8])

v = np.array([[1,  1], [1,  0], [1, -1], [0,  1], [0,  0],
              [0, -1], [-1,  1], [-1,  0], [-1, -1]])
t = np.array([1/36, 1/9, 1/36, 1/9, 4/9, 1/9, 1/36, 1/9, 1/36])


def iterate_lbm_modelling(
    geo, v0, dens_init, max_iter, Re, nx=96, ny=96, uLB=0.04, return_step=10, omega=2):
```

```
solid = geo == 1


vel = np.zeros((2, nx, ny))
vel[0, 0, :] = v0


f_prev = equilibrium(dens_init, vel)


for step in range(max_iter):


    f_prev[col3, -1, :] = f_prev[col3, -2, :]


    dens, macro_vel = macroscopic(f_prev)


    macro_vel[:, 0, :] = vel[:, 0, :]
    dens[0, :] = 1/(1-macro_vel[0, 0, :]) * (np.sum(f_prev[col2, 0, :], axis=0) +
                                2*np.sum(f_prev[col3, 0, :], axis=0))


    try:
        res = corrected_velocity_with_dens(
            macro_vel.copy(), dens, solid, geo, 1, 1)
    except ValueError as e:
        print("Unable to correct velocity, end modelling.")
        break
```

```
        macro_vel, pressure_field, b_part = res['new_vel'], res['p_without_bc'], res['b']


        feq = equilibrium(dens, macro_vel)
        f_prev[[0, 1, 2], 0, :] = feq[[0, 1, 2], 0, :] + \
            f_prev[[8, 7, 6], 0, :] - feq[[8, 7, 6], 0, :]



        f_new = f_prev - omega * (f_prev - feq)



        for i in range(9):
            f_new[i, solid] = f_prev[8-i, solid]


        for i in range(9):
            f_prev[i, :, :] = np.roll(
                np.roll(f_new[i, :, :], v[i, 0], axis=0),
                v[i, 1], axis=1)
        if step % return_step == 0:
            yield step, b_part.copy(), dens.copy(), pressure_field.copy()



def macroscopic(f_prev, nx=96, ny=96):
```

```
    dens = np.sum(f_prev, axis=0)
    macro_vel = np.zeros((2, nx, ny))
    for i in range(9):
        macro_vel[0, :, :] += v[i, 0] * f_prev[i, :, :]
        macro_vel[1, :, :] += v[i, 1] * f_prev[i, :, :]
    macro_vel /= dens
    return dens, macro_vel


def equilibrium(dens, macro_vel, nx=96, ny=96):
    usqr = 3/2 * (macro_vel[0]**2 + macro_vel[1]**2)
    feq = np.zeros((9, nx, ny))
    for i in range(9):
        cu = 3 * (v[i, 0]*macro_vel[0, :, :] + v[i, 1]*macro_vel[1, :, :])
        feq[i, :, :] = dens*t[i] * (1 + cu + 0.5*cu**2 - usqr)
    return feq


def generate_coef_arr_from_dens(r_sum, ra, rb, rc, rd):
    rows, cols = r_sum.shape
    coef_arr = np.zeros((rows*cols, rows*cols))
    ix1 = np.arange(rows*cols)
    coef_arr[ix1, ix1] = r_sum.ravel()
```

```
    ix_a = ix1 + cols
    ix_b = ix1 - cols
    ix_c = ix1 + 1
    ix_d = ix1 - 1
    for ix, r in zip([ix_a, ix_b, ix_c, ix_d], [ra, rb, rc, rd]):
        m = (ix >= 0) & (ix < rows*cols)
        coef_arr[ix1[m], ix[m]] = -r.ravel()[ix[m]]
    return coef_arr


def make_ground_truth_with_dens(f, r_sum, ra, rb, rc, rd, geo, grid_size=96):
    A_coef = generate_coef_arr_from_dens(r_sum, ra, rb, rc, rd)
    fr, fc = np.where(geo == 0)
    ix = fr * grid_size + fc
    # A_coef = A_coef[ix, :]
    # A_coef = A_coef[:, ix]
    A_coef = A_coef[np.ix_(ix, ix)]
    # A_coef = np.round(A_coef, 6)
    A = csr_matrix(A_coef)
    ml = pyamg.ruge_stuben_solver(A)
    u = np.zeros(f.shape)
    f_v = f[fr, fc]
    u_v = ml.solve(f_v, tol=1e-11)
```

```
    u[fr, fc] = u_v
    return u


# 0 - x, 1 - y (vel)
def corrected_velocity_with_dens(
        vel, dens, obs, geo, dx, dt, p_bc=0):
    dens_padded = np.pad(dens, ((1, 1), (1, 1)), 'edge')
    ra = 2 / (dens_padded[1:-1, 1:-1] + dens_padded[2:, 1:-1])
    rb = 2 / (dens_padded[1:-1, 1:-1] + dens_padded[:-2, 1:-1])
    rc = 2 / (dens_padded[1:-1, 1:-1] + dens_padded[1:-1, 2:])
    rd = 2 / (dens_padded[1:-1, 1:-1] + dens_padded[1:-1, :-2])

    r_sum = ra + rb + rc + rd

    vx = vel[0, :, :].copy()
    vy = vel[1, :, :].copy()
    vx[obs] = 0
    vy[obs] = 0
    vx_padded = np.pad(vx, ((1, 1), (1, 1)), 'edge')
    vy_padded = np.pad(vy, ((1, 1), (1, 1)), 'edge')
    vdx = vx_padded[2:, 1:-1] - vx_padded[:-2, 1:-1]
    vdy = vy_padded[1:-1, 2:] - vy_padded[1:-1, :-2]
```

```
    b = -(vdx+vdy)/(2*dx*dt)
    p = make_ground_truth_with_dens(b, r_sum, ra, rb, rc, rd, geo)
    p_old = p.copy()
    p[0, :] = p_bc
    p_padded = np.pad(p, ((1, 1), (1, 1)), 'edge')
    pdx = p_padded[2:, 1:-1] - p_padded[:-2, 1:-1]
    pdy = p_padded[1:-1, 2:] - p_padded[1:-1, :-2]
    new_vel = vel.copy()
    new_vel[:, obs] = 0
    new_vx = new_vel[0, :, :] - dt*pdx/(2*dens*dx)
    new_vy = new_vel[1, :, :] - dt*pdy/(2*dens*dx)
    new_vel[0, :, :] = new_vx
    new_vel[1, :, :] = new_vy
    return {'new_vel': new_vel, 'p': p, 'b': b, 'pdx': pdx, 'pdy': pdy,
            'p_without_bc': p_old}
```

Лістинг А.5 – модель poisson_cnn.py

```
from tensorflow import keras
from tensorflow.keras.layers import Input, Add
from tensorflow.keras.layers import Conv2D, UpSampling2D, AveragePooling2D
from tensorflow.keras.layers import Concatenate, BatchNormalization
from tensorflow.keras.layers import ReLU, Activation
```

```
from tensorflow.keras.layers import MaxPooling2D, ZeroPadding2D

from tensorflow.keras.initializers import glorot_uniform

from tensorflow.keras.models import Model

def fpn_geo_poisson_with_rho_extended(field_side=96, version=None):
    resnet_poisson_backbone = resnet_geo_poisson_with_density_v2(
        field_side=field_side)
    f_in, dist_arr_in = resnet_poisson_backbone.input
    rho_arr_in = Input((field_side, field_side, 1), name='rho_arr')
    C1, C2, C3, C4, C5 = resnet_poisson_backbone.outputs
    P1 = reduce_resnet_output(C1)  # 96
    P2 = reduce_resnet_output(C2)  # 48
    P3 = reduce_resnet_output(C3)  # 24
    P4 = reduce_resnet_output(C4)  # 12
    P5 = reduce_resnet_output(C5)  # 6

    P5_up = UpSampling2D(size=(16, 16))(P5)
    P4_up = UpSampling2D(size=(8, 8))(P4)
    P3_up = UpSampling2D(size=(4, 4))(P3)
    P2_up = UpSampling2D(size=(2, 2))(P2)
```

```
    concat_0 = Concatenate(axis=3)([P5_up, P4_up, P3_up, P2_up, P1])

    # concat_with_input = Concatenate(axis=3)([P4_up, P3_up, P2_up, P1])

    smooth_conv_1 = Conv2D(32, 3, padding='same')(concat_0)
    concat_1 = Concatenate(axis=3)([smooth_conv_1, rho_arr_in])

    smooth_conv_2 = Conv2D(16, 3, padding='same')(concat_1)
    concat_2 = Concatenate(axis=3)([smooth_conv_2, rho_arr_in])

    smooth_conv_3 = Conv2D(8, 3, padding='same')(concat_2)
    concat_3 = Concatenate(axis=3)([smooth_conv_3, rho_arr_in])

    output = Conv2D(1, 3, padding='same', activation='tanh')(concat_3)

    name = name = f'fpn_geo_poisson_with_rho_{field_side}'
    if version is not None:
        name += f'_v{version}'
    return Model(inputs=[f_in, dist_arr_in, rho_arr_in], outputs=output, name=name)


def reduce_resnet_output(output, feature_size=32):
    return Conv2D(feature_size, kernel_size=1,
```

```
                strides=1, padding='same')(output)


def resnet_geo_poisson_with_density_v2(field_side=96, is_pre_conv=True):
    # Define the input as a tensor with shape input_shape
    F_input = Input((field_side, field_side, 1), name='RHS')
    Dist_arr_input = Input((field_side, field_side, 1), name='dist_arr')
    # Density_arr_input = Input((field_side, field_side, 1), name='rho_arr')
    X_input = Concatenate(axis=-1)([
        F_input, Dist_arr_input])


    # Zero-Padding


    if is_pre_conv:
        X = Conv2D(64, (5, 5), strides=(1, 1), padding='same',
                   kernel_initializer=glorot_uniform(seed=0),
                   name='pre_conv')(X_input)
        X = BatchNormalization(axis=3, name='pre_conv_bn')(X)
        X = Activation('relu')(X)
    else:
        X = X_input


    C1 = X
```

```
# S1
X = conv_block(X, kernel_size=3, filters=[
        32, 32, 64], stride=2, stage=1, block='a')
X = identity_block(X, 3, [32, 32, 64], stage=1, block='b')
X = identity_block(X, 3, [32, 32, 64], stage=1, block='c')

C2 = X
# S2
X = conv_block(X, kernel_size=3, filters=[
        48, 48, 96], stride=2, stage=2, block='a')
X = identity_block(X, 3, [48, 48, 96], stage=2, block='b')
X = identity_block(X, 3, [48, 48, 96], stage=2, block='c')

C3 = X
# S3
X = conv_block(X, kernel_size=3, filters=[
        64, 64, 128], stride=2, stage=3, block='a')
X = identity_block(X, 3, [64, 64, 128], stage=3, block='b')
X = identity_block(X, 3, [64, 64, 128], stage=3, block='c')

C4 = X

# S4
```

```
    X = conv_block(X, kernel_size=3, filters=[
                64, 64, 128], stride=2, stage=4, block='a')
    X = identity_block(X, 3, [64, 64, 128], stage=4, block='b')
    X = identity_block(X, 3, [64, 64, 128], stage=4, block='c')

    C5 = X

    model = Model(inputs=[F_input, Dist_arr_input],
                outputs=[C1, C2, C3, C4, C5],
                name='resnet_poisson_with_rho')
    return model

def conv_block(X, kernel_size: int, filters: list, stride: int, stage: int,
               block: str):
    if len(filters) != 3:
        raise ValueError('Filters len must be 3')
    conv_name = 'conv_conv_' + str(stage) + block
    bn_name = 'conv_bn_' + str(stage) + block
    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value
    X_shortcut = X
```

```
# First component of main path
X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(stride, stride),
           padding='valid', kernel_initializer=glorot_uniform(seed=0),
           name=f'{conv_name}_1')(X)
X = BatchNormalization(axis=3, name=f'{bn_name}_1')(X)
X = Activation('relu')(X)

# Second component of main path
X = Conv2D(filters=F2, kernel_size=(kernel_size, kernel_size),
           strides=(1, 1), padding='same',
           kernel_initializer=glorot_uniform(seed=0),
           name=f'{conv_name}_2')(X)
X = BatchNormalization(axis=3, name=f'{bn_name}_2')(X)
X = Activation('relu')(X)

# Third component of main path
X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1),
           padding='valid', kernel_initializer=glorot_uniform(seed=0),
           name=f'{conv_name}_3')(X)
X = BatchNormalization(axis=3, name=f'{bn_name}_3')(X)

# shortcut
```

```
    X_shortcut = Conv2D(filters=F3, kernel_size=(1, 1),
                        strides=(stride, stride), padding='valid',
                        kernel_initializer=glorot_uniform(seed=0),
                        name=f'{conv_name}_short')(X_shortcut)
    X_shortcut = BatchNormalization(axis=3)(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X


def identity_block(X, kernel_size, filters, stage: int, block: str):
    conv_name = 'ident_conv_' + str(stage) + block
    bn_name = 'ident_bn_' + str(stage) + block
    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value
    X_shortcut = X

    # First component of main path
    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(1, 1), padding='valid',
```

```
            kernel_initializer=glorot_uniform(seed=0),
            name=f'{conv_name}_1')(X)
    X = BatchNormalization(axis=3, name=f'{bn_name}_1')(X)
    X = Activation('relu')(X)

    # Second component of main path
    X = Conv2D(filters=F2, kernel_size=(kernel_size, kernel_size), strides=(
        1, 1), padding='same',  kernel_initializer=glorot_uniform(seed=0),
        name=f'{conv_name}_2')(X)
    X = BatchNormalization(axis=3, name=f'{bn_name}_2')(X)
    X = Activation('relu')(X)

    # Third component of main path
    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid',
            kernel_initializer=glorot_uniform(seed=0),
            name=f'{conv_name}_3')(X)
    X = BatchNormalization(axis=3, name=f'{bn_name}_3')(X)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)
```

Лістинг А.6 – модуль lbm_method.py

```
from time import time as time_now
import json

from numpy import *
import numpy as np
import pandas as pd

v = array([[1,  1], [1,  0], [1, -1], [0,  1], [0,  0],
          [0, -1], [-1,  1], [-1,  0], [-1, -1]])
t = array([1/36, 1/9, 1/36, 1/9, 4/9, 1/9, 1/36, 1/9, 1/36])

def macroscopic(fin):
    dens = sum(fin, axis=0)
    macro_vel = zeros((2, dens.shape[0], dens.shape[1]))
    for i in range(9):
        macro_vel[0, :, :] += v[i, 0] * fin[i, :, :]
        macro_vel[1, :, :] += v[i, 1] * fin[i, :, :]
    macro_vel /= dens
    return dens, macro_vel

def equilibrium(dens, macor_vel):              # Equilibrium distribution function.
    usqr = 3/2 * (macor_vel[0]**2 + macor_vel[1]**2)
    feq = zeros((9, macor_vel[0].shape[0], macor_vel[0].shape[1]))
```

```
    for i in range(9):
        cu = 3 * (v[i, 0]*macor_vel[0, :, :] + v[i, 1]*macor_vel[1, :, :])
        feq[i, :, :] = dens*t[i] * (1 + cu + 0.5*cu**2 - usqr)
    return feq



def equilibrium_2(dens, macro_vel):
    ax = np.sqrt(3*macro_vel[0]**2 + 1)
    ay = np.sqrt(3*macro_vel[1]**2 + 1)
    b1 = (2 - ax)*(2 - ay)
    ex = (2*macro_vel[0] + ax) / (1 - macro_vel[0])
    ey = (2*macro_vel[1] + ay) / (1 - macro_vel[1])
    feq = zeros((9, macro_vel[0].shape[0], macro_vel[0].shape[1]))
    for i in range(9):
        t1 = 1
        t2 = 1
        if v[i, 0] == 1:
            t1 = ex
        elif v[i, 0] == -1:
            t1 = 1/ex

        if v[i, 1] == 1:
            t2 = ey
```

```
        elif v[i, 1] == -1:
            t2 = 1/ey
        pr = b1 * t1 * t2
        feq[i, :, :] = dens*t[i] * pr
    return feq


def make_modelling(geo, plot_folder, num, omega=2, denso=1000, v0 = 0.001, maxIter=1500,
                   input_direction = 'top', output_direction = 'bottom',
                   save_fields=False, is_correct_vel=False, is_alternative_eq=True):
    nx, ny = geo.shape[0], geo.shape[1]


    obstacle = geo == 1
    vel = zeros((2, nx, ny))
    if input_direction == 'top':
        vel[0, 0, :] = v0
    elif input_direction == 'right':
        vel[1, :, -1] = -v0
    if is_alternative_eq:
        f_prev = equilibrium_2(denso, vel)
    else:
        f_prev = equilibrium(denso, vel)
```

```
vels = []

denss = []

pressures = []

data = []


for time in range(maxIter):

    st = time_now()


    # Right wall: outflow condition.

    if output_direction == 'top':

        f_prev[[0,1,2], 0, :] = f_prev[[0,1,2], 1, :]

    elif output_direction == 'bottom':

        f_prev[[6, 7, 8], -1, :] = f_prev[[6, 7, 8], -2, :]


    dens, macro_vel = macroscopic(f_prev)


    if input_direction == 'top':

        macro_vel[0, 0, :] = v0

        dens[0, :] = 1/(1-macro_vel[0, 0, :]) * (sum(f_prev[[3, 4, 5], 0, :], axis=0) +

                                2*sum(f_prev[[6, 7, 8], 0, :], axis=0))

    elif input_direction == 'right':

        macro_vel[1, :, -1] = -v0

        dens[:, -1] = 1/(1+macro_vel[1, :, -1]) * (sum(f_prev[[1, 4, 7], :, -1], axis=0) +
```

```
                2*sum(f_prev[[0, 3, 6], :, -1], axis=0))
if save_fields:
    vels.append(macro_vel.copy())
    denss.append(dens.copy())
if is_correct_vel:

    res = corrected_velocity_with_dens_by_cnn(
          macro_vel.copy(), dens, obstacle, geo, 1, 1, p_bc_side=input_direction)

    macro_vel, pressure_field = res['new_vel'], res['p']
    if save_fields:
       pressures.append(pressure_field.copy())

mean_dens = dens[~obstacle].mean()
std_dens = dens[~obstacle].std()



if is_alternative_eq:
    feq = equilibrium_2(dens, macro_vel)
else:
    feq = equilibrium(dens, macro_vel)

if input_direction == 'right':
```

```
        f_prev[[2, 5, 8], :, -1] = feq[[2, 5, 8], :, -1] + \
            f_prev[[6, 3, 0], :, -1] - feq[[6, 3, 0], :, -1]
    elif input_direction == 'top':
        f_prev[[0, 1, 2], 0, :] = feq[[0, 1, 2], 0, :] + \
            f_prev[[8, 7, 6], 0, :] - feq[[8, 7, 6], 0, :]

    f_new = f_prev - omega * (f_prev - feq)

    for i in range(9):
        f_new[i, obstacle] = f_prev[8-i, obstacle]

    for i in range(9):
        f_prev[i, :, :] = roll(
            roll(f_new[i, :, :], v[i, 0], axis=0),
            v[i, 1], axis=1)
    data.append({'iter': time, 'mean_dens': mean_dens, 'std_dens': std_dens})

df = pd.DataFrame(data)
df.to_csv(plot_folder / f'meta.csv')
if save_fields:
    with open(plot_folder / 'vels.npy', 'wb') as f:
        np.save(f, np.array(vels))
    with open(plot_folder / 'denss.npy', 'wb') as f:
```

```
            np.save(f, np.array(denss))
        with open(plot_folder / 'pressures.npy', 'wb') as f:
            np.save(f, np.array(pressures))


    meta_data = {'v0': v0, 'omega': omega, 'num': num, 'denso': denso, 'is_correct_vel': is_correct_vel,
                 'max_iter': maxIter, }
    with open(plot_folder / 'meta.json', 'w') as f:
        json.dump(meta_data, f)


def corrected_velocity_with_dens_by_cnn(vel, dens, obs, distance_arr, dx, dt, p_bc = 0, field_size=96,
                                        smooth_output=False, sigma=1):
    vx = vel[0,:,:].copy()
    vy = vel[1,:,:].copy()
    vx[obs] = 0
    vy[obs] = 0
    vx_padded = np.pad(vx, ((1,1), (1,1)), 'edge')
    vy_padded = np.pad(vy, ((1,1), (1,1)), 'edge')
    vdx = vx_padded[2:,1:-1] - vx_padded[:-2,1:-1]
    vdy = vy_padded[1:-1,2:] - vy_padded[1:-1,:-2]
    b = -(vdx+vdy)/(2*dx*dt)
```

```
dens_k = max(np.abs(dens).max(), 1)
dens_cnn_input = dens / dens_k

b_k = max(np.abs(b).max(), 1)
b_cnn_input = b / b_k

field = predict(b_cnn_input, distance_arr, dens_cnn_input, obs, field_size=field_size,
            smooth_output=smooth_output, sigma=sigma)
p = field * b_k * dens_k

p[obs] = 0
p[0, :] = p_bc
p_padded = np.pad(p, ((1,1), (1,1)), 'edge')
pdx = p_padded[2:,1:-1] - p_padded[:-2,1:-1]
pdy = p_padded[1:-1,2:] - p_padded[1:-1,:-2]
new_vel = vel.copy()
new_vel[:, obs] = 0
new_vx = new_vel[0,:,:] - dt*pdx/(2*dens*dx)
new_vy = new_vel[1,:,:] - dt*pdy/(2*dens*dx)
new_vel[0,:,:] = new_vx
new_vel[1,:,:] = new_vy
return {'new_vel': new_vel, 'p': p, 'b': b,
        'pdx': pdx, 'pdy': pdy,
```

```
        'b_cnn_input': b_cnn_input,
        'dens_cnn_input': dens_cnn_input,
        'cnn_output': field}


def predict(model, b_cnn_input, dist_arr, dens_cnn_input, qt, min_max, transform = True, field_size=96):
    pred = model.predict([b_cnn_input[np.newaxis, ...], dist_arr[np.newaxis, ...], dens_cnn_input[np.newaxis, ...]])
    pred = pred.reshape((field_size, field_size))
    if not transform:
        return pred
    field = qt.inverse_transform(min_max.inverse_transform(pred.reshape(-1, 1))).reshape(field_size, field_size)
    return field
```

ДОДАТОК Б

Список публікацій здобувача

1. Novotarskyi M.A., Stirenko S.G., Gordienko Y.G., **Kuzmych V.A.** Deep reinforcement learning with sparse distributed memory for “Water World” problem solving //Radio Electronics, Computer Science, Control.– 2021.– № 1.– P.136-143. DOI 10.15588/1607-3274-2021-1-14
2. **Kuzmych V.A.**, Novotarskyi M.A., Nesterenko O.B. Solving Poisson Equation with Convolutional Neural Networks // “Radio Electronics, Computer Science, Control” – 2022.– № 1. p. 48-57. DOI 10.15588/1607-3274-2022-1-6
3. **Valentyn Kuzmych**, Mykhailo Novotarskyi, Accelerating simulation of the PDE solution by the structure of the convolutional neural network modifying, Lecture Notes on Data Engineering and Communications Technologies 135, p. 3-15 ISSN 23674512 DOI 10.1007/978- 3-031-04809-8 (Scopus)
4. М.А.Новотарський Дворівневий метод моделювання руху рідини за допомогою решітчастої моделі Больцмана та згорткової нейронної мережі / М.А. Новотарський, **В.А. Кузьмич** // Електронне моделювання – Т45, № 5 – Київ, 2023. -С. 39-53

Праці наукових конференцій:

1. **Kuzmych V.**, Novotarskyi M. “Application of machine learning in the modeling of physical processes” // International Conference ICSFTI2020, May 14-15, 2020. Зроблено доповідь
2. **Kuzmych V.**, Novotarskyi M. Solving Poisson Equation with Convolutional Neural Networks // International Conference ICSFTI2021, May 12-13, 2021. Зроблено доповідь.
3. **Valentyn Kuzmych**, Mykhailo Novotarskyi, Accelerating simulation of the PDE solution by the structure of the convolutional neural network modifying, The 2nd International Conference on Artificial Intelligence and Logistics Engineering (ICAILE2022) - Virtual Conference, 20-22 February 2022. Зроблено доповідь.

4. **Kuzmych V.**, Novotarskyi M. Simulation of fluid motion in closed surfaces using a lattice Boltzmann model // International Conference ICSFTI2022, June 30, 2022. Зроблено доповідь.