DOI: https://doi.org/10.36910/6775-2524-0560-2025-60-29
УДК 004.94
Руксов Євгеній Вікторович, аспірант
https://orcid.org/0009-0003-3741-1647
Мороз Борис Іванович, д.т.н., професор
https://orcid.org/0000-0002-5625-0864
Національного технічного університету «Дніпровська політехніка», м. Дніпро, Україна

ФУНКЦІОНАЛЬНЕ ПРЕДСТАВЛЕННЯ 3D-ОБ'ЄКТІВ, ЯК МЕТОД УНІВЕРСАЛІЗАЦІЇ ДАНИХ В ГЕНЕРАТИВНИХ МОДЕЛЯХ МАШИННОГО НАВЧАННЯ

Руксов Є. В., Мороз Б. І. Функціональне представлення 3D-об'єктів, як метод універсалізації даних в генеративних моделях машинного навчання. У статті пропонується новий метод представлення тривимірних об'єктів через скалярні функції векторного аргументу. Цей спосіб представлення дозволяє звести функцію до ряду нормалізованих векторів, що є ключовою вимогою під час побудови моделей машинного навчання класу Transformer. Необхідність розробки такого методу ґрунтується на загальній концептуальній гіпотезі про можливість використання на поточний момент найбільш успішного типу генеративних нейронних мереж, який є центральним в сфері великих мовних моделей (LLM), для генерування 3D-об'єктів із специфічними фізичними властивостями. Ця гіпотеза випливає з того, що моделі класу Transformer (зокрема GPT) вже були застосовні для генерування двовимірних зображень, тому за допомогою правильного представлення тривимірних форм об'єктів можливо розширити застосування до 3D-моделювання. Результати тестування запропонованого методу показали високу точність згортання «хмари точок» поверхні 3D-моделі у ряд векторів (ембедингів) та відновлення з цього ряду вихідної фігури. Демонстраційна реалізація методу має недолік, пов'язаний з неточністю інтерполяції функції, проте це не є обмеженням самого методу, а лише його технічної реалізації. Справжнє обмеження методу полягає у звуженні до лише одного класу топологічних просторів. Подальші дослідження будуть спрямовані на узагальнення цього методу на всі можливі типи топологій.
Ключові слова: 3D-моделювання, генеративні моделі ШІ, функціональне представлення форми об'єкта, група подібності фігури, топологічна поверхня нульового роду, швидке перетворення Фур'є.

Ruksov Y., Moroz B. Functional representation of 3D objects as a method of data generalization in generative machine learning models. The article proposes a new method for representing three-dimensional objects through scalar functions of a vector argument. This method of representation allows reducing the function to a series of normalized vectors, which is a key requirement for building machine learning models of Transformer class. The need to develop such a method is based on a general conceptual hypothesis about the possibility of using the currently most successful type of generative neural networks, which is central in the field of large language models (LLM), to generate 3D objects with specific physical properties. This hypothesis follows from the fact that Transformer-like models (in particular, GPT) have already been applied to generate two-dimensional images, therefore, with correct representation of 3D object's shapes, it is possible to extend their application to 3D modeling. The results of testing the proposed method showed high accuracy in collapsing the "point cloud" of the 3D model surface into a series of vectors (embeddings) and restoring the original figure from this series. The demonstrational implementation of the method has the drawback of inaccuracy in function interpolation, but this is not a limitation of the method itself, but only of its technical implementation. The real limitation of the method is its narrowing down to only one class of topological spaces. Further research will be aimed at generalizing this method to all possible types of topologies.
Keywords: 3D modeling, generative AI models, functional representation of object shape, figure similarity group, topological surface of zero genus, fast Fourier transform.

Постановка наукової проблеми. Будь-який виріб, який під час експлуатації піддається впливу навколишнього середовища, має певні властивості, які є динамічними та залежать від поточного стану та загальних властивостей навколишнього середовища. Під час проектування будови різних виробів необхідно враховувати різні аспекти впливу навколишнього середовища. Для багатьох типів фізичного впливу середовища на виріб характер залежності між параметрами (які є кількісним відображенням певних корисних властивостей) виробу та параметрами середовища є нелінійним та має складну багатовимірну сутність. Для такого типу залежності важко встановити однозначне пряме правило, за яким можна передбачити зміни параметрів. Тому існує потреба пошуку методів, які дозволять хоча б апроксимувати функціональну залежність між зміною стану середовища та зміною параметрів виробу.

Корисні властивості аеродинамічних виробів пов'язані із впливом динамічного в'язкого середовища (газу) на об'єкт. Характер цього впливу змінюється в залежності від певних початкових умов самого середовища, а також від просторових властивостей самого об'єкту впливу. Весь цей набір умов визначають разом із загальними принципами динаміки руху частинок, з яких складається аеродинамічне середовище, можуть звузити пошук функціональних закономірностей. Весь математичний апарат аеродинамічних процесів зосереджений в рівняннях Нав'є-Стокса, нелінійних диференціальних рівняннях в частинних похідних (ДРЧП), тому виходячи з цих рівнянь можна

встановити, на який саме елемент цієї фізичної системи може вплинути конструктор аеродинамічних виробів під час моделювання будови виробу.

Серед базових параметрів аеродинамічного виробу можна виділити:

- дальність польоту;
- вантажомісткість;
- максимальна швидкість польоту;
- максимальна висота польоту;
- маневреність і т.д.

Більшість цих характеристик мають багатофакторний зв'язок з різними аспектами навколишнього середовища. Основні фізичні параметри, які випливають саме з аеродинаміки, це:

1. Коефіцієнт підйомної сили (Lift Coefficient, C_L)

$$C_L = \frac{L}{\frac{1}{2}\rho V^2 S}$$

де L – підйомна сила (в Ньютонах), ρ – густина повітря (кг/м³), V – швидкість відносного обтікання (м/с), S – характерна площа.

2. Коефіцієнт лобового опору (Drag Coefficient, C_D)

$$C_D = \frac{D}{\frac{1}{2}\rho V^2 S}$$

де D – сила опору (в Ньютонах).

3. Аеродинамічна якість – C_L/C_D.

Примітка: для простоти викладу інформації про доцільність зміни підходу представлення комп'ютерних моделей об'єктів у вступній частині буде використано поняття «форма» (англійською shape) об'єкта, що в геометрії відповідає класу подібності, а в теорії груп – орбіті фігури (як елемента множини всіх фігур) відносно груп перетворень: зсуву, обертання, масштабування в координатному просторі.

Обчислення значень цих параметрів під час моделювання робиться зазвичай за допомогою чисельних методів розв'язання диференціальних рівнянь в частинних похідних із заданими крайовими та початковими умовами. Крайові умови є тими, які напряму залежать від форми виробу. Основними фізичними параметрами середовища, які необхідно визначити є поля тиску та швидкості потоку навколо крайових умов. Саме вони безпосередньо впливають на зазначені вище параметри виробу. Тому загальний ланцюг зв'язків параметрів виглядає приблизно так: форма виробу → крайові умови рівняння Нав'є-Стокса → поле значень тиску та швидкості → коефіцієнти лобового опору та підйомної сили → шукані значення параметрів виробу. Тобто ключовим аспектом, який є об'єктом моделювання конструктора аеродинамічних виробів та відіграє ключову роль при пошуку оптимальної конструкції виробу, – це форма (shape) об'єкта.

В CFD системах зазвичай використовуються певний клас чисельних методів для обчислення значень тиску та швидкості потоку. Серед найбільш ефективних та універсальних (з точки зору різноманітності підтримуваних форм досліджуваного об'єкта) чисельних методів найпопулярнішим є FEM (Finite element method). Багатообіцяючим є напрямок досліджень, в яких розробляють способи інтеграції FEM в нейронні мережі типу PINN (physics-informed neural networks). Як продемонстрували автори дослідження [1], реалізація цього концепту дозволила значно знизити витрати на тренування таких моделей та збільшити точність обчислень. Безумовно, для різних типів ДРЧП існують різні оптимізовані чисельні методи, проте наразі необхідно розробити загальний концепт інтеграції найбільш фундаментальних чисельних методів (а FEM лежить в основі більшості сучасних вузькоспеціалізованих методів) з моделями машинного навчання задля розширення можливостей симулювання фізичних процесів в умовах браку або зашумлення початкових даних чи іншої специфічної природи досліджуваних фізичних процесів. Цей етап розвитку ще не пройшов, цілісної концепції прикладних моделей машинного навчання для вирішення фізичних задач все ще немає, тому такі роботи як [1] мають високу цінність.

Для того, щоб інтегрувати форму об'єкта в постановку крайової задачі треба мати таке представлення форми (shape), яке буде складатися з дискретних елементів. Зазвичай це polygon mesh (у найбільш простому вигляді triangle mesh), який є фактично геометричною реалізацією симпліційного комплексу (спеціальний тип топологічних просторів) [2].

Для рівняння Нав'є-Стокса необхідно знати лише крайові координати об'єкта, внутрішня його структура не має значення, адже визначаються параметри навколишнього середовища, тобто об'єктом аналізу є в'язке середовище, а не сам об'єкт. Таким чином для коректної постановки крайової задачі умова Діріхле буде визначена на всій замкненій поверхні об'єкта, і це буде достатньо для вирішення задачі.

Таким чином, необхідно розробити таке представлення форми об'єкта, яке було б достатньо узагальненим для всіх точок поверхні, тобто для побудови множини всіх точок поверхні достатньо знати лише загальний принцип їх розподілу. Це дозволить «запакувати» форму поверхні в однозначне представлення, абстрагуючись від кожної окремої точки поверхні. Зрештою, з'явиться можливість змістити «увагу» генеративної моделі безпосередньо до форми (shape), а не окремих точок на ній.

Таким чином, це представлення буде мати цілісну інформацію про кривизну поверхні об'єкта, яка не обтяжена ані внутрішньою структурою, ані координатами кожної окремої точки. Для вирішення поставленої задачі пошуку оптимальної форми (shape) за певними числовими характеристиками необхідно перш за все побудувати з інформації про форми (shape) певний простір, кожна точка якого буде мати свої координати. Пошук підходящої точки в цьому просторі є ціллю задачі оптимізації форми (shape) за певними кількісними показниками, які залежать виключно від характеру кривизни поверхні.

Недолік точкової генерації (найбільш просунутою наразі генеративною моделлю цього класу можна вважати MSG-Voxel-GAN, описана в статті [3]) полягає в тому, що така генерація працює з многовидами на локальному рівні, не маючи інформації про загальну форму (shape) та топологію множини. Функціональний зв'язок (який апроксимує генеративна модель) між множиною точок у фізичному просторі та фізичними параметрами середовища не є прямим, більше того він навіть не є однорідним відносно типу елементів множин цього відношення. Тобто множина точок – це простір дискретних точок в декартовій системі координат, при тому точка в такому просторі є лише бінарним скаляром (наявність чи відсутність тіла у відповідній координаті). Фізичні параметри середовища описуються функцією, яка визначається явним або неявним методом з простору цих функцій, який описаний відповідним диференціальним рівнянням. Тобто точкою в такому просторі є функція. Таким чином через різну природу просторів апроксимація такого відношення функціонального простору на простір бінарних скалярів може бути занадто складною навіть для генеративного машинного навчання.

Першим кроком для вирішення задачі апроксимації відношення між множинами є встановлення однорідності цих множин. Прирівняти функціональний простір фізичного закону до простору бінарних скалярів призведе до втрати інформації про фізику досліджуваного процесу. Тож залишається лише звести множину точок поверхні до функціонального простору. Очевидно, що кривизна поверхні має функціональну природу, тож це і є ключовою аксіомою нового методу представлення форми (shape).

Функціональне представлення форми, опис методу. Спершу треба дати чітке формальне визначення форми (shape) об'єкта. Об'єкт в цьому контексті є фактично геометричною фігурою. Тоді з елементарної геометрії можна виділити теорему перетворення подібності, з якої можна дати визначення класу подібності, що і є формою (shape) фігури.

Клас подібності в геометрії – це множина всіх фігур, які є подібними. А подібність фігур визначається збереженням чисельного значення всіх кутів фігури, при операціях перенесення (зсуву по координатам), повороту та масштабуванні над фігурою.

Можна розширити клас фігур, які складаються не лише з прямих ліній, а й з гладких кривих чи навіть поверхонь в тривимірному просторі R^3. В такому разі при відповідних операціях над фігурою буде збережено диференціальні властивості кривих, які складають фігуру. Наразі не будемо вдаватися в детальні з цієї точки зору, а змістимо акцент на самі операції.

Маючи операції над певними математичними об'єктами, можна перейти з геометричної парадигми в алгебраїчну, а саме роздивитися поняття класу подібності з точки зору теорії груп. Найбільш точно та водночас достатньо абстрактно властивості геометричних об'єктів відносно дій певних груп перетворення описує теорія геометричних інваріант, сформульована Девідом Мумфордом (остання редакція його книги [4]).

З операцій перенесення, повороту, масштабування можна побудувати групу подібності фігур:

$$Sim(n) = \mathbb{R}^n \rtimes (O(n) \times \mathbb{R}_{>0}) \quad (1)$$

де n – вимір просторових координат.

З цього можна визначити поняття еквівалентності фігур F та G по відношенню до групи подібності Sim(n). Дія групи на фігуру позначається оператором «·» .

$$F \sim G \Leftrightarrow \exists g \in Sim(n): G = g \cdot F \quad (2)$$

Якщо зібрати всі фігури, які є еквівалентними одна одній згідно заданого формального визначення, то утвориться орбіта фігури відносно групи подібності Sim(n).

$$\mathcal{O}(F) = \{g \cdot F \mid g \in Sim(n)\} \quad (3)$$

Це, власне, і є еквівалент геометричному класу подібності в теорії груп. Тобто, форма (shape) фігури = клас подібності = орбіта фігури відносно групи подібності Sim(n).

Для зручності подальших розрахунків варто побудувати фактор-простір фігур на основі групи подібності. Шлях до факторизації простору фігур проходить через «згортання» орбіти фігури до однієї точки фактор-простору. Для цього треба обрати з групи її стабілізатор (тобто таку операцію, яка при дії на фігуру буде давати цю ж фігуру). Далі за допомогою косетів можна довести ізоморфізм орбіти фігури дії всіх косетів, які побудовані на основі нормальної підгрупи симетрій (які і є стабілізатором). Але деталі цього доведені будуть опущені наразі [5]. Зрештою за допомогою цих механізмів можна звести простір всіх фігур до фактор-простору фігур відносно групи подібності Sim(n), де одна точка простору відповідає цілій орбіті фігури. Таким чином ми дали визначення простору форм (shape) фігур.

На практиці, існує механізм вибору представника орбіти фігури, а саме калібрування. Для цього потрібно відібрати «калібр» орбіти F_{std} (або стандартного представника орбіти, який відповідає ряду умов для кожної складової групи подібності Sim(n)):

1. Для зсуву – центр мас фігури в нулі координат.
2. Для повороту – найближча до центру мас точка поверхні направлена вздовж координатної орти *i* (вісь *Ox*).

3. Для масштабу – розмір повинен бути таким, щоб найближча до центру мас точка знаходилась на відстані 1 від центру мас.

Таким чином, для кожного елемента всієї множини фігур існує свій єдиний калібрувальний елемент групи подібності Sim(n), який «стандартизує» фігуру для подальших операцій з нею. Пошук таких калібрувальних перетворень не розглядається в рамках цього дослідження, тому це питання буде опущено наразі (багато практичних прикладів калібрування орбіт груп різних перетворень можна знайти в огляді [6]).

Всі ці висновки через парадигму теорії груп дозволяють відкинути зайву інформацію про просторові характеристики фігури, сфокусувавши увагу виключно на формі (shape) фігури. Далі відкривається шлях до побудови функціонального представлення точки фактор-простору фігур.

Зрештою, з функціонального представлення можна буде побудувати координатну систему, по якій кожна фігура буде визначатися вектором дійсних чисел (що інколи називаються «ембедингом»). Але це буде занадто складне та багатовимірне представлення.

Проте можна спростити таке представлення, розбивши його на послідовність нормованих по розмірності векторів, які відображають якийсь елемент функціонального представлення. Сума цих окремих елементів буде давати загальну фігуру. А кількість доданків суми може бути необмежена. Таким чином можливо побудувати представлення у вигляді ряду.

Для сучасних генеративних моделей машинного навчання ряди векторів є найбільш ефективним способом подання генерованої інформації, з точки зору якості побудови надскладних апроксимацій відношень. Найпотужніші генеративні моделі наразі показали свою високу ефективність в сфері генерування тексту. Людська мова – це нелінійне складне переплетіння абстрактних понять через асоціації та відношення. Генеративні моделі навчилися апроксимувати такі складні відношення за допомогою генерування рядів та застосування таких потужних механізмів, як «увага» [7].

Для генерування таких же складних апроксимацій, як відношення однієї функції до іншої, методи, які застосовуються у великих мовних моделях (LLM), вбачаються органічними – тож, ідеологічною гіпотезою цього дослідження буде саме ця теза. Але ця стаття присвячена лише теоретизації однієї підгіпотези, а саме: будь-яку фігуру з ненульовою площею (об'ємом) можна

представити у вигляді ряду нормованих векторів, які при певній комбінації нелінійних тривіальних операцій можуть бути складені у функціональну форму, за якою можна відновити всю множину точок фігури.

Замкнені поверхні, які утворюють геометричні фігури, непросто виразити через прямі аналітичні формули. Адже такі функції мають чіткі обмеження щодо області визначення та області значення функції, при тому для кожної фігури ці обмеження мають різні значення і калібрування не допоможе в нормалізації цих значень. Для реалізації повного відновлення всієї множини точок поверхні фігури для будь-якої точки фактор-простору фігур відносно групи подібності Sim(n) з функціонального представлення необхідна універсальна для всіх фігур область визначення функції (тобто скінченний діапазон можливих значень аргументу функції, який можна в подальшому дискретизувати для чисельних методів).

Параметрична форма подання функції, де параметром є кут в сферичних координатах, відповідає наведеним вище критеріям. Діапазон можливих значень параметру є універсальним і лежить між 0 та 2π. Параметричну форму подання функції можна ще трактувати як вектор-значну функцію. При тому для двовимірного простору аргументом є скаляр з R, а значення – вектор з R^2; для тривимірного простору аргумент функції – це вже вектор з R^2, а значення – вектор з R^3. Тобто в загальному вигляді це можна записати так:

$$f\colon \mathbb{R}^{n-1} \to \mathbb{R}^n \tag{4}$$

Проте для поставленої задачі достатньо обмежитись тривимірним простором. Зручним є ще такий запис:

$$f(\theta,\varphi) = \begin{cases} X(\theta,\varphi) \\ Y(\theta,\varphi), \\ Z(\theta,\varphi) \end{cases} \qquad \theta \in [0,2\pi], \varphi \in [0,\pi] \tag{5}$$

Для топологічної поверхні нульового роду (тобто гомеоморфної сфері без дірок) функції X, Y, Z можна звести до такого вигляду:

$$f(\theta,\varphi) = \begin{cases} r(\theta,\varphi)\sin\theta\cos\dfrac{\varphi}{2} \\ r(\theta,\varphi)\sin\theta\sin\dfrac{\varphi}{2}, \\ r(\theta,\varphi)\cos\theta \end{cases} \qquad \theta \in [0,2\pi], \varphi \in [0,2\pi] \tag{6}$$

Це представлення виходить з параметричного рівняння сфери, але радіус (тобто відстань від нуля координат до точки на сфері) є змінною величиною, яка залежить від комбінації кутів θ та φ. Також для зручності був зміщений діапазон значень кута φ, щоб область була квадратною.

Таким чином, необхідно знайти функцію r(θ, φ) визначену на квадраті θ, φ ∈ [0, 2π] (позначимо область як Ω), при тому значення в кожній точці на краях квадрату Ω повинні бути попарно рівними значенням в точках на протилежному краї квадрату (тобто цей квадрат Ω можна «склеїти» краями у фігуру, гомеоморфну тору). Формально цю умову можна сформулювати так: фактор-простір ділянки поверхні Ω, яка описується функцією r(θ, φ), повинен бути тор (тобто топологічна поверхня першого роду). Ця умова є ключовою для забезпечення цілісності та замкненості досліджуваної фігури при відновленні множини точок її поверхні за функцією (6). Ще однією умовою для цієї функції є додатність значення на всій області визначення, тобто r(θ, φ) > 0 на всій області Ω. Формально умови записуються так:

$$\forall(\theta,\varphi) \in \Omega : r(\theta,\varphi) > 0, \qquad \Omega = [0,2\pi] \times [0,2\pi] \subset \mathbb{R}^2 \tag{7.1}$$

$$q\colon \Omega \longrightarrow T^2, \qquad q(\theta,\varphi) = (\theta \; mod \; 2\pi, \varphi \; mod \; 2\pi), T^2 - torus \tag{7.2}$$

Ще одне обмеження полягає в тому, що в загальному випадку r(θ, φ) є насправді мультифункцією, тобто множина значень цієї функції складається не з одиничних скалярів, а із

скінченних непустих підмножин R. Тому для запропонованого методу наразі буде введено два обмеження (які в подальших дослідженнях будуть подолані більш складними алгоритмами):

1. Фігури повинні бути топологічними поверхнями першого роду.
2. Функція r(θ, φ) з параметричного представлення фігури повинна бути звичайною функцією, значення якої є одиничний скаляр.

Умова 7.2 дозволяє класифікувати функцію r(θ, φ) як періодичну. Також, очевидно, ця функція може бути розкладена в Гільбертовому просторі 2π періодичних функцій в ортогональний базис синусів та косинусів, тобто розкладена в ряд Фур'є. Таким чином для кожної функції r(θ, φ) можна обчислити її координати в цьому ортогональному базисі. При тому ця функція є неперервною на ділянці Ω, що додає прихильності до розкладання саме в ряд Фур'є.

Для реальних обчислювальних задач приходиться працювати зі скінченними наборами точок функції, тобто з дискретизованими функціями. За умови достатньо великої частоти дискретизації (так щоб в крок не потрапляли суттєві «перепади» функції) можна навіть уникнути спотворень під час відновлення фігури з ряду Фур'є. Це твердження випливає з теореми відліків Найквіста-Шеннона. Більше того, буде достатньо обчислювати лише часткову суму ряду Фур'є, що дозволить ефективно реалізувати алгоритм в програмі, тим паче вже існує швидкі методи обчислення коефіцієнтів ряду, зокрема відомий FFT алгоритм отримав багато покращень та оптимізаційних модифікацій. Серед найновіших нововведень був запропонований TurboFFT [8] для виконання обчислень на базі графічних процесорів з приростом продуктивності у 300% в порівнянні з попереднім cuFFT, як зазначали автори дослідження.

Використання рядів Фур'є дозволить представити складну функцію r(θ, φ) у вигляді ряду координат, або ряду векторів. Це відкриває можливість застосовувати генеративні моделі машинного навчання для побудови форми (shape) фігури, застосовуючи найрозвиненіші на даний момент методи з класу великих мовних моделей (LLM).

Тепер деталізуємо розкладання в ряд Фур'є двовимірної періодичної функції. Такий ряд виглядає так (тригонометрична форма, часткова сума):

$$\begin{aligned} g(\theta,\varphi) = \frac{a_{00}}{4} + \frac{1}{2}\sum_{h=1}^{H} a_{h0}\cos h\theta + \frac{1}{2}\sum_{l=1}^{L} a_{0l}\cos l\varphi \\ + \sum_{h=1}^{H}\sum_{l=1}^{L}(a_{hl}\cos h\theta\cos l\varphi + b_{hl}\cos h\theta\sin l\varphi + c_{hl}\sin h\theta\cos l\varphi \\ + d_{hl}\sin h\theta\sin l\varphi) \end{aligned} \tag{8}$$

Як бачимо, на кожну гармоніку ряду Фур'є для двовимірних функцій необхідно обчислити по 4 коефіцієнти. Алгоритм обчислення цих коефіцієнтів не буде описаний тут, як правило, для таких задач користуються швидким перетворенням Фур'є (FFT) [8]. Складність FFT алгоритму для двовимірної функції при рівномірній квадратній сітці дискретизації дорівнює $O(N^2 * \log_2 N)$, крім того, існує багато оптимізованих реалізацій цього алгоритму, які дозволяють збільшити рівень паралелізму обчислень – тому FFT є підходящим методом обчислення координат функції в ортогональному базисі ряду Фур'є. Зазначимо лише загальну аналітичну формулу для обчислення коефіцієнтів ряду:

$$a_{hl} = \frac{1}{\pi^2}\sum_{j}^{N}\sum_{i}^{N} g(\theta_j,\varphi_i)\cos h\theta_j\cos l\varphi_i \tag{9.1}$$

$$b_{hl} = \frac{1}{\pi^2}\sum_{j}^{N}\sum_{i}^{N} g(\theta_j,\varphi_i)\cos h\theta_j\sin l\varphi_i \tag{9.2}$$

$$c_{hl} = \frac{1}{\pi^2}\sum_{j}^{N}\sum_{i}^{N} g(\theta_j,\varphi_i)\sin h\theta_j\cos l\varphi_i \tag{9.3}$$

$$d_{hl} = \frac{1}{\pi^2} \sum_{j}^{N} \sum_{i}^{N} g(\theta_j, \varphi_i) \sin h\theta_j \sin l\varphi_i \qquad (9.4)$$

В звичайній версії перетворення Фур'є існує ймовірність отримати від'ємні значення функції, що заборонено умовою (7.1). Тому для гарантування адекватності апроксимації можна застосувати техніку попереднього логорифмування функції, а при відновленні множини точок поверхні обчислювати експоненту кожного значення. Це дозволить уникнути від'ємних значень при відновленні фігури, щоб не спотворювати її топологію. Таким чином, функція g(θ, φ) з виразу (8) відноситься до функції r(θ, φ) таким чином:

$$g(\theta, \varphi) = \ln (r(\theta, \varphi)) \qquad (10.1)$$

$$r(\theta, \varphi) = \exp (g(\theta, \varphi)) \qquad (10.2)$$

Перша формула застосовується під час зведення до функціонального представлення форми (shape) фігури, а друга – під час відновлення до масиву точок поверхні.

Як бачимо з формул (9) повний набір координат можна представити у вигляді матриці, елементи якої – чотиривимірні вектори. Не будемо називати це тензором (тривимірною матрицею), адже кожен елемент матриці в проміжних обчисленнях за FFT алгоритмом представляє собою одне комплексне число, а чотиривимірний вектор утворюється зі складових дійсної та уявної частин комплексного числа. Форму цієї матриці для спрощення обрано квадратну, тобто повинна виконуватися умова H=L (з формули (8)). При тому розмірність матриці є довільною, тобто генератор може синтезувати координатну матрицю будь-якого розміру (звісно для сучасних генераторів векторних рядів необхідно все ж мати обмеження довжини ряду, так зване вікно).

Існують моделі машинного навчання, які засновані на генеративних моделях типу GPT та можуть генерувати вектори (ембединги) у формі матриці. Це некласичне застосування моделей, які зазвичай використовуються для генерування тексту, в сфері генерування зображень. Модель ImageGPT від OpenAI [9] здатна синтезувати зображення, використовуючи механізми та переваги методів «уваги», які дозволяють моделі, враховуючи всі попередні токени, передбачати наступні. Для зображень це працює так: матриця перетворюється у вектор фіксованої довжини, а генератор разом з передбаченням наступного токена також формує для нього індексацію в матриці. Тобто є певний окремий ряд індексів (номери стовпців та рядків) для токенів, і для зображень важливо зберігати одну роздільну здатність генерованих зображень, тому ряд векторів завжди має фіксовану довжину і примусово зупиняється генерація при досягненні останнього пікселя зображення.

Для генерації функціонального представлення фігури необхідно прибрати фіксацію розміру матриці, при тому треба зберігати квадратну форму без пробілів генерації. Тобто необхідно навчати модель генерувати стоп-сигнал (EOS – end of sequence) і лише для певних індексів токена. А саме лише кутові токени на головній діагоналі матриці можуть бути кінцевими (тобто лише після них можуть генеруватися EOS).

Для збереження цілісної квадратної форми матриці порядок генерованих токенів можна застосувати такий алгоритм:

1. Додається новий стовпець такої ж довжини, як попередній.
2. Додається новий рядок такої ж довжини, як попередній.
3. Додається кутовий елемент діагоналі (лише на цьому етапі може бути зупинка генерації).

Зрештою така генерація може бути розроблена як модифікація класичної моделі ImageGPT. За даними статті [9] ця модель показала свою високу ефективність на різних бенчмарках генераторів зображень, тому з інтеграцією такої моделі у функціональне представлення тривимірного об'єкта є великі перспективи для створення повноцінної генеративної моделі для об'ємних фігур.

Результати. Для демонстрації адекватності запропонованого методу представлення тривимірних моделей розроблена програма на мові Python з функціями візуалізації.

Основні задачі демонстраційної програми:

1. Показати можливість зведення тривимірної моделі об'єкта до матриці комплексних коефіцієнтів, та зрештою до матриці чотиривимірних векторів для тригонометричної форми.

2. Показати можливість відновлення тривимірної моделі об'єкта з матриці координат (коефіцієнтів) ряду Фур'є.

Далі надано лістинг коду:

```
import matplotlib.pyplot as plt
import numpy as np
import open3d as o3d
from pxr import Usd, UsdGeom
from scipy.interpolate import griddata
from scipy.spatial.transform import Rotation
def extract_points_from_usd(file_path):
    stage = Usd.Stage.Open(file_path)
    points_list = []
    for prim in stage.Traverse():
        if prim.IsA(UsdGeom.Mesh):
            mesh = UsdGeom.Mesh(prim)
            points = mesh.GetPointsAttr().Get()
            for p in points:
                points_list.append((p[0], p[1], p[2]))
    return np.array(points_list)
def normalize_points_array(points_array):
    center = np.mean(points_array, axis=0)
    centered_points = points_array - center
    distances = np.linalg.norm(centered_points, axis=1)
    closest_point = centered_points[np.argmin(distances)]
    v = closest_point / np.linalg.norm(closest_point)
    z_axis = np.array([0, 0, 1])
    rot_axis = np.cross(v, z_axis)
    rot_angle = np.arccos(np.clip(np.dot(v, z_axis), -1.0, 1.0))
    rot_axis = rot_axis / np.linalg.norm(rot_axis)
    rot = Rotation.from_rotvec(rot_angle * rot_axis)
    rotated_points = rot.apply(centered_points)
    distances = np.linalg.norm(rotated_points, axis=1)
    min_dist = np.min(distances)
    scale_factor = 1.0 / min_dist
    scaled_points = rotated_points * scale_factor
    return scaled_points
def cartesian_to_spherical(points_array):
    x = points_array[:, 0]
    y = points_array[:, 1]
    z = points_array[:, 2]
    r = np.sqrt(x ** 2 + y ** 2 + z ** 2)
    theta = np.arccos(z / r)
    phi = np.arctan2(y, x)
    theta = 2 * theta
    phi = phi + np.pi
    return np.stack((theta, phi, r), axis=1)
def spherical_to_cartesian(spherical_array):
    theta = spherical_array[:, 0] / 2
    phi = spherical_array[:, 1] - np.pi
    r = spherical_array[:, 2]
    x = r * np.sin(theta) * np.cos(phi)
    y = r * np.sin(theta) * np.sin(phi)
    z = r * np.cos(theta)
    return np.stack((x, y, z), axis=1)
def add_periodic_padding_percent(points_array, period_x=2 * np.pi, period_y=2 * np.pi, percent=10.0):
```

```
    dx = (percent / 100.0) * period_x
    dy = (percent / 100.0) * period_y
    x = points_array[:, 0]
    y = points_array[:, 1]
    x_min_mask = x < np.min(x) + dx
    x_max_mask = x > np.max(x) - dx
    y_min_mask = y < np.min(y) + dy
    y_max_mask = y > np.max(y) - dy
    strips = [points_array[x_min_mask].copy()]
    strips[-1][:, 0] += period_x
    strips.append(points_array[x_max_mask].copy())
    strips[-1][:, 0] -= period_x
    strips.append(points_array[y_min_mask].copy())
    strips[-1][:, 1] += period_y
    strips.append(points_array[y_max_mask].copy())
    strips[-1][:, 1] -= period_y
    corners = [points_array[x_min_mask & y_min_mask].copy()]
    corners[-1][:, 0] += period_x
    corners[-1][:, 1] += period_y
    corners.append(points_array[x_min_mask & y_max_mask].copy())
    corners[-1][:, 0] += period_x
    corners[-1][:, 1] -= period_y
    corners.append(points_array[x_max_mask & y_min_mask].copy())
    corners[-1][:, 0] -= period_x
    corners[-1][:, 1] += period_y
    corners.append(points_array[x_max_mask & y_max_mask].copy())
    corners[-1][:, 0] -= period_x
    corners[-1][:, 1] -= period_y
    return np.vstack([points_array] + strips + corners)
def interpolate_to_regular_grid(spherical_array, grid_res=100, method='cubic', fill_value=np.nan):
    spherical_array = add_periodic_padding_percent(spherical_array)
    x, y, z = spherical_array[:, 0], spherical_array[:, 1], spherical_array[:, 2]
    xi = np.linspace(0, 2 * np.pi, grid_res, endpoint=False)
    yi = np.linspace(0, 2 * np.pi, grid_res, endpoint=False)
    xi, yi = np.meshgrid(xi, yi)
    zi = griddata((x, y), z, (xi, yi), method=method, fill_value=fill_value)
    return [xi, yi, zi]
def grid_to_point_array(grid):
    x_flat = grid[0].flatten()
    y_flat = grid[1].flatten()
    z_flat = grid[2].flatten()
    points_array = np.stack((x_flat, y_flat, z_flat), axis=1)
    return points_array[~np.isnan(points_array).any(axis=1)]
def fft_transform(spherical_array):
    grid = interpolate_to_regular_grid(spherical_array, fill_value=1)
    F = np.fft.fft2(grid[2])
    F_shifted = np.fft.fftshift(F)
    return F_shifted
def ifft_transform(F):
    F_unshifted = np.fft.ifftshift(F)
    return np.fft.ifft2(F_unshifted).real
def complex_to_vector_spectrum(F_shifted):
    F = np.fft.ifftshift(F_shifted)
    Ny, Nx = F.shape
    F = F / (Nx * Ny)
    idx = lambda k, N: k % N
```

```
    A = np.zeros_like(F, dtype=float)
    B = np.zeros_like(F, dtype=float)
    C = np.zeros_like(F, dtype=float)
    D = np.zeros_like(F, dtype=float)
    for m in range(Nx):
        for n in range(Ny):
            c1 = F[idx(+m, Nx), idx(+n, Ny)]
            c2 = F[idx(+m, Nx), idx(-n, Ny)]
            c3 = F[idx(-m, Nx), idx(+n, Ny)]
            c4 = F[idx(-m, Nx), idx(-n, Ny)]
            A[m, n] = (c1 + c2 + c3 + c4).real
            B[m, n] = (1j * (c1 - c2 + c3 - c4)).real
            C[m, n] = (1j * (c1 + c2 - c3 - c4)).real
            D[m, n] = (-c1 + c2 + c3 - c4).real
    return np.stack((A, B, C, D), axis=-1)
def vector_to_complex_spectrum(V):
    F_norm = 1 / 4 * (V[:, :, 0] - 1j * V[:, :, 1] - 1j * V[:, :, 2] - V[:, :, 3])
    Ny, Nx = F_norm.shape
    F_unshifted = F_norm * (Nx * Ny)
    return np.fft.fftshift(F_unshifted)
def plot_surface_from_grid(grid, title="3D Surface", x_title="X", y_title="Y", z_title="Z"):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(grid[0], grid[1], grid[2], cmap='viridis', linewidth=0, antialiased=True)
    ax.set_title(title)
    ax.set_xlabel(x_title)
    ax.set_ylabel(y_title)
    ax.set_zlabel(z_title)
    plt.tight_layout()
    plt.show()
def plot_surface_from_spherical_points(spherical_array, title="3D Surface"):
    grid = interpolate_to_regular_grid(spherical_array, fill_value=1)
    plot_surface_from_grid(grid, title)
def plot_fft_magnitude_3d(F):
    Ny, Nx = F.shape
    X, Y = np.meshgrid(np.linspace(-Nx // 2, Nx // 2, Nx), np.linspace(-Ny // 2, Ny // 2, Ny))
    plot_surface_from_grid([X, Y, np.log1p(F)], "3D Visualization of FFT Magnitude",
                           "Frequency X", "Frequency Y", "|F(kx, ky)|")
def visualize_4d_tensor_components(tensor, cmap='coolwarm'):
    H, W, _ = tensor.shape
    fig, axes = plt.subplots(2, 2, figsize=(16, 16))
    titles = np.array([['A (cos·cos)', 'B (cos·sin)'], ['C (sin·cos)', 'D (sin·sin)']])
    k = 0
    for i in range(2):
        for j in range(2):
            channel = tensor[:, :, k]
            im = axes[i, j].imshow(channel, cmap=cmap)
            axes[i, j].set_title(titles[i, j])
            axes[i, j].axis('off')
            fig.colorbar(im, ax=axes[i, j], shrink=0.8)
            k += 1
    plt.tight_layout()
    plt.show()
def plot_with_open3d(points_array):
    pc = o3d.geometry.PointCloud()
    pc.points = o3d.utility.Vector3dVector(points_array)
```

```
    axes = o3d.geometry.TriangleMesh.create_coordinate_frame(size=1.0, origin=[0, 0, 0])
    o3d.visualization.draw_geometries([pc, axes])
points = extract_points_from_usd("data/model.usdc")
points = normalize_points_array(points)
spherical_points = cartesian_to_spherical(points)
plot_surface_from_spherical_points(spherical_points)
complex_spectrum = fft_transform(spherical_points)
plot_fft_magnitude_3d(complex_spectrum)
vector_spectrum = complex_to_vector_spectrum(complex_spectrum)
visualize_4d_tensor_components(vector_spectrum)
complex_spectrum_reconstructed = vector_to_complex_spectrum(vector_spectrum)
Z_reconstructed = ifft_transform(complex_spectrum_reconstructed)
Z_reconstructed[Z_reconstructed < 1] = 1
spherical_grid = interpolate_to_regular_grid(spherical_points)
spherical_grid[2] = Z_reconstructed
spherical_points_reconstructed = grid_to_point_array(spherical_grid)
plot_surface_from_spherical_points(spherical_points_reconstructed)
points_reconstructed = spherical_to_cartesian(spherical_points_reconstructed)
spherical_points = grid_to_point_array(interpolate_to_regular_grid(spherical_points))
plot_with_open3d(spherical_points)
plot_with_open3d(spherical_points_reconstructed)
plot_with_open3d(points)
plot_with_open3d(points_reconstructed)
```

Візьмемо для прикладу просту сфероподібну 3D-модель (подано два ракурси на рис. 1).

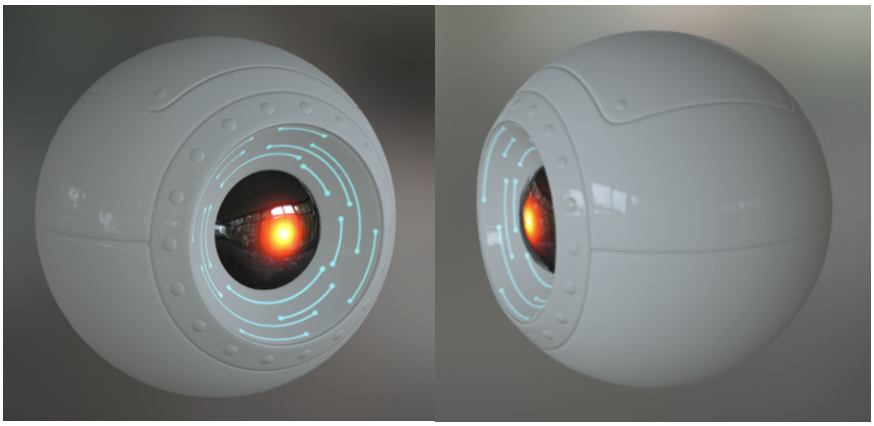

Рис. 1. Приклад простої 3D-моделі

Задля демонстрації оберненості перетворень будуть надаватися проміжні етапи обчислень.

Відкинувши зайві елементи та конвертувавши у формат .usdc можна скористатися наданим тестовим алгоритмом. З набору точок в тривимірному просторі отримаємо функцію $r(\theta, \varphi)$. На рис. 2 надана ця функція на етапі перетворення у матрицю векторів та на етапі відновлення з матриці векторів назад у тривимірну модель.

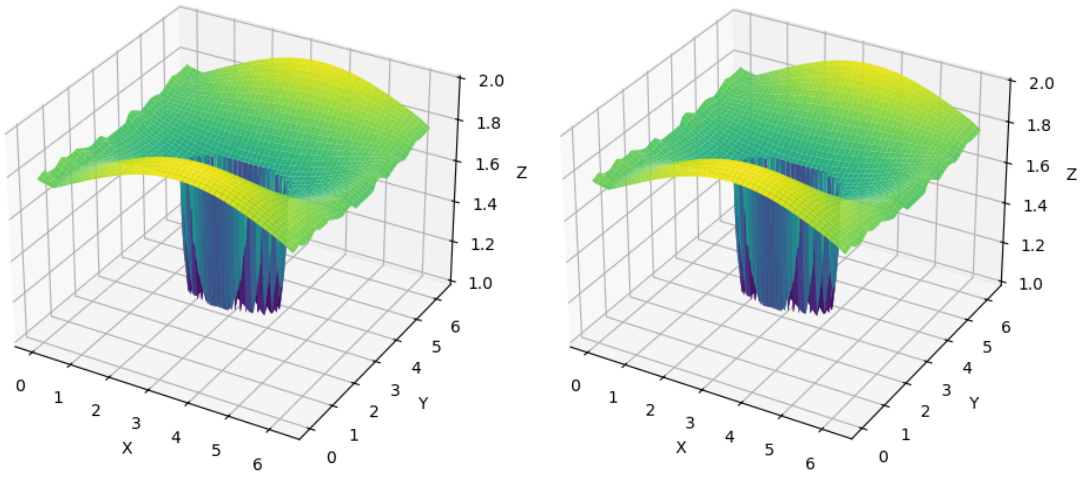

Рис. 2. Зліва: функція r(θ, φ) при перетворенні в матрицю векторів. Справа: відновлена функція r(θ, φ) з матриці векторів

Результат перетворення Фур'є можна представити у вигляді тривимірної карти коефіцієнтів гармонік (зі змішенням малих частот всередину карти). Зображено на рис. 3.

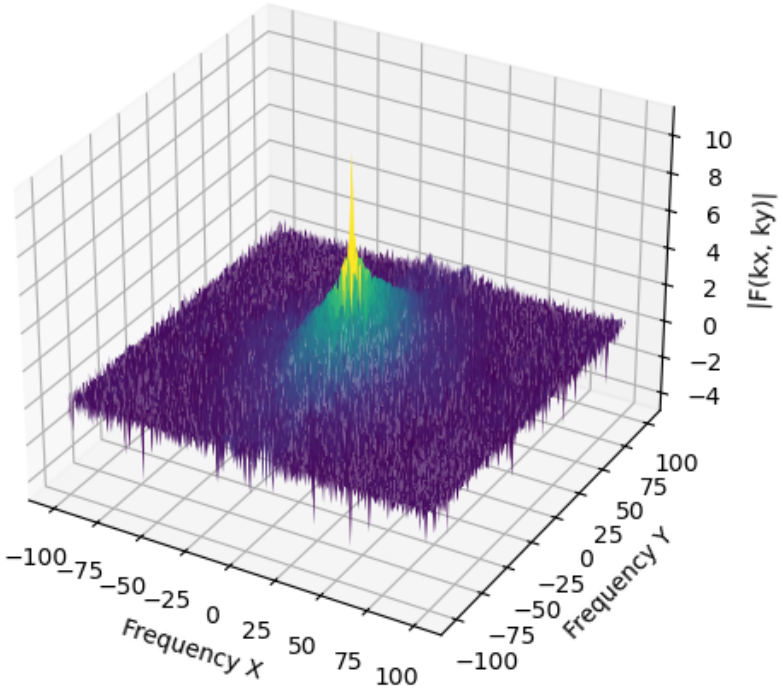

Рис. 3. Карта коефіцієнтів гармонік

Для повної картини розкладемо, отриману в результаті розкладання функції в двовимірний ряд Фур'є, матрицю векторів на чотири канали. Візуалізація наведена на рис. 4.

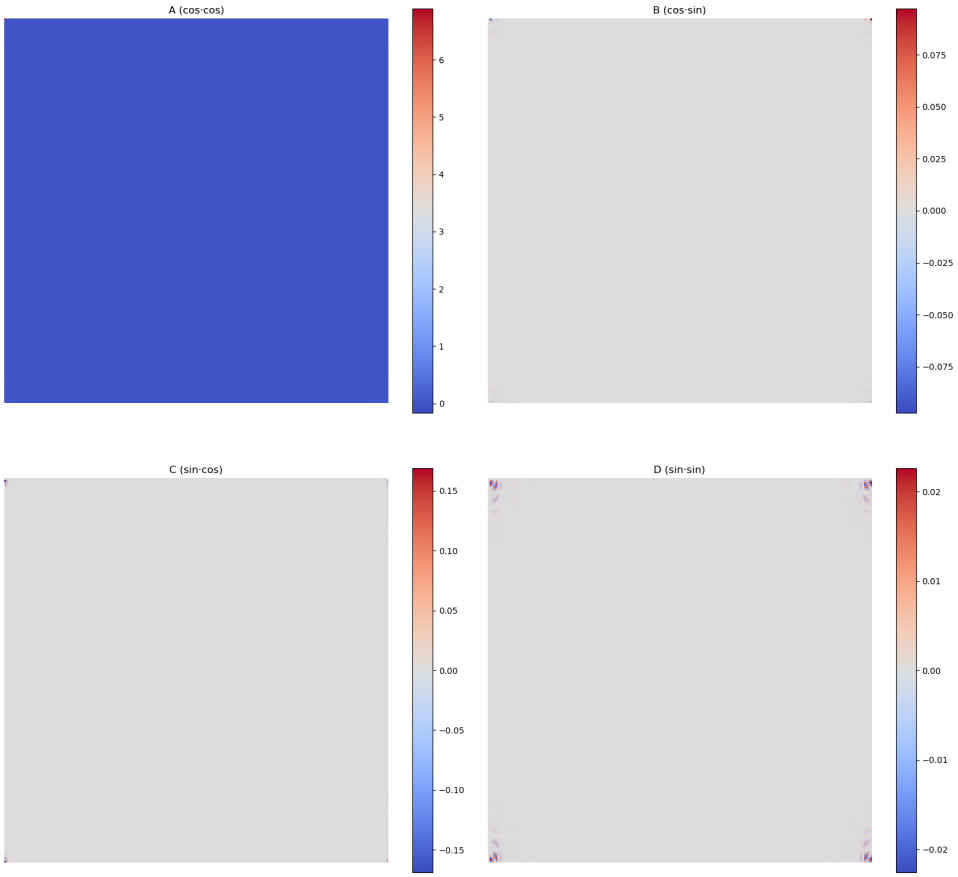

Рис. 4. Візуалізація матриці чотиривимірних векторів

Для демонстрації якості запропонованого методу функціонального представлення тривимірної фігури наведемо порівняння оригінальної «хмари точок» фігури та відновленої форми фігури з матриці векторів. Зображено на рис. 5.

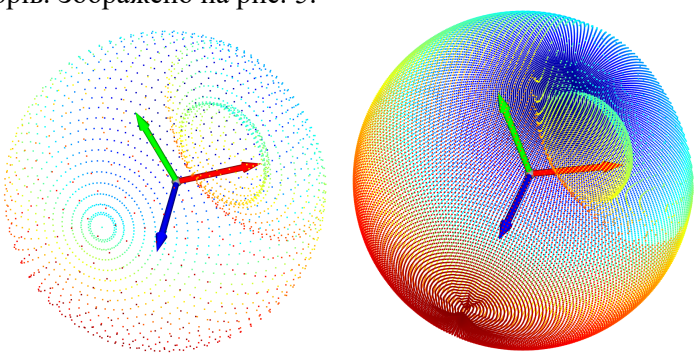

Рис. 5. Зліва: оригінальна «хмара точок». Справа: відновлена з матриці векторів

При порівнянні можна відмітити дві особливості:

1. Значний рівень інтерполяції точок, що обумовлено необхідністю проведення нормалізації набору точок на регулярній сітці задля більш точної апроксимації функції рядом Фур'є.

2. В критичних точках помітні артефакти, що є насправді наслідком недосконалої кубічної інтерполяції (інші стандартні методи інтерполяції також тестувалися, але не дали значних покращень). Техніка штучного розширення набору точок (для періодичних функцій це легко зробити) була застосована задля мінімізації артефактів та спотворень на краях діапазону значень аргументів функції $r(\theta, \varphi)$. Проте повністю позбутися цих дефектів не вдалось, тому в майбутньому варто розробити спеціальний метод інтерполяції функцій, який буде забезпечувати потреби запропонованого механізму функціонального представлення 3D-фігур.

Висновки та перспективи подальшого дослідження. У даній статті було розглянуто існуючі на поточний момент підходи до моделювання аеродинамічних виробів. Наразі немає цілісної системи швидкого моделювання (генерування) виробів із специфічними фізичними параметрами.

Тому на основі аналізу найкращих сучасних методів та засобів генерування структурованої інформації була висунута концептуальна гіпотеза про можливість ефективного застосування методів зі сфери LLM для синтезу тривимірних фігур певної форми, яка задовольняла б певним фізичним характеристикам. Цю гіпотезу було розділено на підгіпотези. Ця стаття була сфокусована не підгіпотезі, яка стверджує, що будь-яку фігуру можна представити у вигляді ряду нормалізованих векторів (що є ключовою умовою для побудови сучасних моделей зі сфери LLM). В даній статті був сформульований та деталізований метод представлення форми фігури, який за результатами експериментів зміг довести істинність висунутої підгіпотези, хоча й з певними обмеженнями.

Подальші дослідження будуть сфокусовані на узагальненні та розширенні запропонованого методу представлення. А наступним етапом має бути дослідження можливостей адаптація сучасних чисельних методів розв'язання ДРЧП для інтеграції їх в повний контур генеративних моделей машинного навчання класу Transformer або GPT (які є ядром LLM).

Список бібліографічного опису:

1. Franziska Griese, Fabian Hoppe, Alexander Rüttgers, Philipp Knechtges. Preconditioned FEM-based neural networks for solving incompressible fluid flows and related inverse problems. – Journal of Computational and Applied Mathematics. 2025. – 15 с. DOI: https://doi.org/10.1016/j.cam.2025.116663.
2. Silvia Biasotti. Computational topology methods for shape modelling applications. – Dissertation, Università degli studi di Genova. 2004. – 194 с. URL: https://core.ac.uk/download/ pdf/37832978.pdf.
3. Bingxu Wang, Jinhui Lan, Feifan Li. MSG-Voxel-GAN: multi-scale gradient voxel GAN for 3D object generation. – Multimedia Tools and Applications. 2024. – 17 с. DOI: https://doi.org/10.1007/s11042-023-17116-9.
4. David Mumford , John Fogarty , Frances Kirwan. Geometric Invariant Theory. – Springer. 1994. – 294 с. URL: https://link.springer.com/book/9783540569633.
5. О.Г.Ганюшкін, О.О.Безущак. Теорія груп. – Київ. Київський національний університет імені Тараса Шевченка. 2005. – 123 с. URL: https://www.mechmat.univ.kiev.ua/wp-content/uploads/ 2018/03/group_l.pdf.
6. Ian L. Dryden, Kanti V. Mardia. Statistical shape analysis: with applications in R. – Wiley Series in Probability and Statistics. 2016. – 496 с. DOI: http://doi.org/10.1002/9781119072492.
7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need. - 31st Conference on Neural Information Processing Systems. 2017. – 11 с. URL: https://surl.lu/qicczb.
8. Shixun Wu, Yujia Zhai, Jinyang Liu, Jiajun Huang, Zizhe Jian, Huangliang Dai, Sheng Di, Franck Cappello, Zizhong Chen. TurboFFT: Co-Designed High-Performance and Fault-Tolerant Fast Fourier Transform on GPUs. – NY, USA. Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2025. – 15 с. DOI: https://doi.org/10.1145/ 3710848.3710853.
9. Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, Ilya Sutskever. Generative Pretraining from Pixels. – International Conference on Machine Learning. 2020. – 12 с. URL: https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf.

References:

1. Franziska Griese, Fabian Hoppe, Alexander Rüttgers, Philipp Knechtges. Preconditioned FEM-based neural networks for solving incompressible fluid flows and related inverse problems. – Journal of Computational and Applied Mathematics. 2025. – 15 p. DOI: https://doi.org/10.1016/j.cam.2025.116663.
2. Silvia Biasotti. Computational topology methods for shape modelling applications. – Dissertation, Università degli studi di Genova. 2004. – 194 p. URL: https://core.ac.uk/download/ pdf/37832978.pdf.
3. Bingxu Wang, Jinhui Lan, Feifan Li. MSG-Voxel-GAN: multi-scale gradient voxel GAN for 3D object generation. – Multimedia Tools and Applications. 2024. – 17 p. DOI: https://doi.org/10.1007/s11042-023-17116-9.
4. David Mumford , John Fogarty , Frances Kirwan. Geometric Invariant Theory. – Springer. 1994. – 294 с. URL: https://link.springer.com/book/9783540569633.
5. Olexandr G. Ganyushkin, Oksana O. Bezushchak. Group theory. – Kyiv. Taras Shevchenko National University of Kyiv. 2005. – 123 p. URL: https://www.mechmat.univ.kiev.ua/wp-content/uploads/ 2018/03/group_l.pdf.
6. Ian L. Dryden, Kanti V. Mardia. Statistical shape analysis: with applications in R. – Wiley Series in Probability and Statistics. 2016. – 496 p. DOI: http://doi.org/10.1002/9781119072492.
7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need. - 31st Conference on Neural Information Processing Systems. 2017. – 11 p. URL: https://surl.lu/qicczb.
8. Shixun Wu, Yujia Zhai, Jinyang Liu, Jiajun Huang, Zizhe Jian, Huangliang Dai, Sheng Di, Franck Cappello, Zizhong Chen. TurboFFT: Co-Designed High-Performance and Fault-Tolerant Fast Fourier Transform on GPUs. – NY, USA. Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2025. – 15 p. DOI: https://doi.org/10.1145/ 3710848.3710853.
9. Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, Ilya Sutskever. Generative Pretraining from Pixels. – International Conference on Machine Learning. 2020. – 12 p. URL: https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf.