

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

## Дипломна робота

Симуляція турбулентності з використанням архітектурного шаблону Entity Component System (ECS)

Виконав: студент групи ПМП-42  
спеціальності  
113 - прикладна математика

Смаль О. В.

(прізвище та ініціали)

Керівник Дяконюк Л. М.

(прізвище та ініціали)

Рецензент \_\_\_\_\_

(прізвище та ініціали)

Львів - 2023

# Зміст

1	Теоретичні відомості	4
1.1	Архітектурний шаблон СКС (система компонента сутність)	4
1.2	Статистична механіка . . . . .	6
1.3	Пружне зіткнення . . . . .	7
1.4	Довжина вільного пробігу . . . . .	9
2	Експерименти	11
2.1	Формулювання задачі . . . . .	11
2.2	Результати . . . . .	11
3	Висновки	16
4	Джерела	17

# Вступ

Зі стрімким розвитком сучасних комп'ютерних технологій та зростаючим попитом на спеціалістів у дотичних галузях, написання справді якісного програмного забезпечення стає прогресивно менш фінансово вигідним. Кожне нове покоління обчислювальних машин нівелює будь-які оптимізації створені для попередніх. Оскільки кінцевий споживач продукту у багатьох випадках не здатен помітити малих відмінностей у швидкості роботи рішень, логічним є жертва їх чистоти та швидкодії задля збільшення швидкості розробки продукту. Така зміна уможливила ріст низькокваліфікованої частини ринку праці інформаційних технологій, що в свою чергу давало змогу задовольняти зростаючий попит ринку. Проте, як і будь-яка стрімко зростаюча економічна конструкція, така фінансова бульбашка приречена на розпад.

Неминучим є своєчасне зменшення об'єму робочої сили розробників програмного забезпечення до кількості яку роботодавці здатні будуть стабільно забезпечувати. Проте, очевидно, така зміна ринку праці призведе до підвищення порогу входу для дотичних професій.

Беручи до уваги всі вищезгадані обставини, має місце думка про пошук альтернативних рішень популярних класів задач тим, які зараз є

широко використовуваними.

У 2007 році команда, що працювала над Operation Flashpoint: Dragon Rising, експериментувала з патерном ECS, а Адам Мартін пізніше написав докладний опис дизайну ECS, включаючи визначення основної термінології та концепцій. Зокрема, робота Мартіна популяризувала ідеї «систем» як елемента першого класу, «сутності як ідентифікатори», «компоненти як необроблені дані» і «код, що зберігається в системах, а не в компонентах або об'єктах». Попри низькорозповсюдженість цього шаблону на час його винайдення, він знайшов застосування у розробці програмного забезпечення що потребувало швидкодії при роботі з великою кількістю об'єктів, такго як ігри та симуляції.

## Мета і предмет дослідження

Метою цієї роботи є дослідження переваг застосування шаблону СКС на прикладі задачі з великою кількістю елементів. Предметом дослідження цієї роботи проводиться порівняння СКС з іншими більш загальноприйнятими шаблонами. За основу експерименту візьмемо симуляцію явища турбулентності, проте описане на рівні традиційної не-статистичної механіки.

# Розділ 1

## Теоретичні відомості

### 1.1 Архітектурний шаблон СКС (система компонента сутність)

Основною ідеєю цього шаблону проектування був чіткий поділ між структурами що відповідали за зберігання даних, та тими що відповідали за роботу з ними:

- Компонента — структура, що обгортає необроблені дані та не несе жодної інформації про логіку їх опрацювання. Для розпізнавання, кожен клас компоненти володіє унікальним ідентифікатором (зазвичай вигляду  $2^n$ ,  $n \in \mathbb{N}$ ,  $n > 0$ , що набуде змісту пізніше). Прикладами компоненти можуть бути розташування об'єкту в просторі чи його швидкість — інформація, що описує одну незалежну властивість об'єкту.
- Сутність (суб'єкта) — структура, що обгортає набір компонент, що

вичерпно описують об'єкт. Окрім компонент, клас структури характеризується сумою ідентифікаторів його компонент. Така агрегація здатна вичерпно описано присутність тої чи іншої компоненти в суб'єкті, оскільки таке представлення ефективно є бітовою множиною. Ця властивість суб'єкти називається архетипом. Прикладом суб'єкти можна назвати частинку, яка в свою чергу матиме компоненти позиції та швидкості.

- Система — структура, що містить бізнес логіку процесу, та ідентифікатор, що визначається як побітне “але” ідентифікаторів всіх класів компонент необхідних для застосування цієї системи. Сумісність пари системи та компоненти визначається як рівність між побітним “і” між архетипом сутності та ідентифікатором системи, тим самим ж ідентифікатором системи.

Стан процесу, описаного за шаблоном СКС, визначається як набір (часто асоціативний масив) масивів сутностей, кожен з яких містить лише один архетип (див. рис. 1.1). Такі масиви називають глибами. Дані масиви створюються з попередньо визначеною ємністю, тому якщо в будь-який момент виконання процесу з'являється необхідність створення більшої кількості сутностей, відповідна глиба розширюється на ще один масив (див. рис. 1.2).

Важливою перевагою СКС є здатність передбачити (навіть, якщо з надлишком), кількість пам'яті необхідної для процесу, і виділити її ще на етапі ініціалізації програми. Це дозволяє зменшити, якщо не припинити повністю, діяльність збирача сміття, таким чином звільняючи додаткові

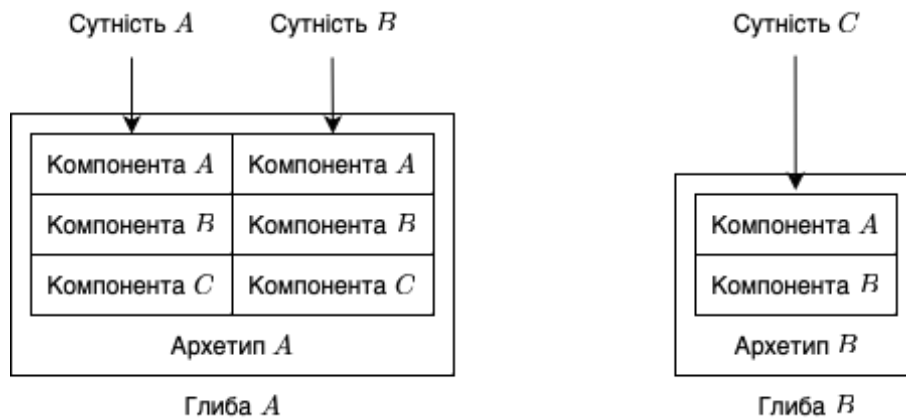


Рис. 1.1: Глиби.

ресурси для програми.

## 1.2 Статистична механіка

Щоб показати різницю між СКС та традиційними підходами до розв’язування задач, розглянемо наступну проблему. Гази першого роду з високою точністю можна описувати як великі за об’ємами колекції сферичних частинок, зіткнення між якими є ідеально пружними. Тому за експеримент розглянемо велику кількість частинок, що пружньо стикаються між собою. Із зростанням кількості симульованих частинок модель прямуватиме до реалістичної, та проявлятиме колективні властивості такі як хвильові явища, турбулентність, кавітація, тощо.

Оскільки власне симуляція не є основною метою даної роботи, далі буде внесено ряд спрощень, що зберігаючи ідею експерименту допомагають уникнути труднощів, відсутність вигоди яких не впливає на результати дослідження.

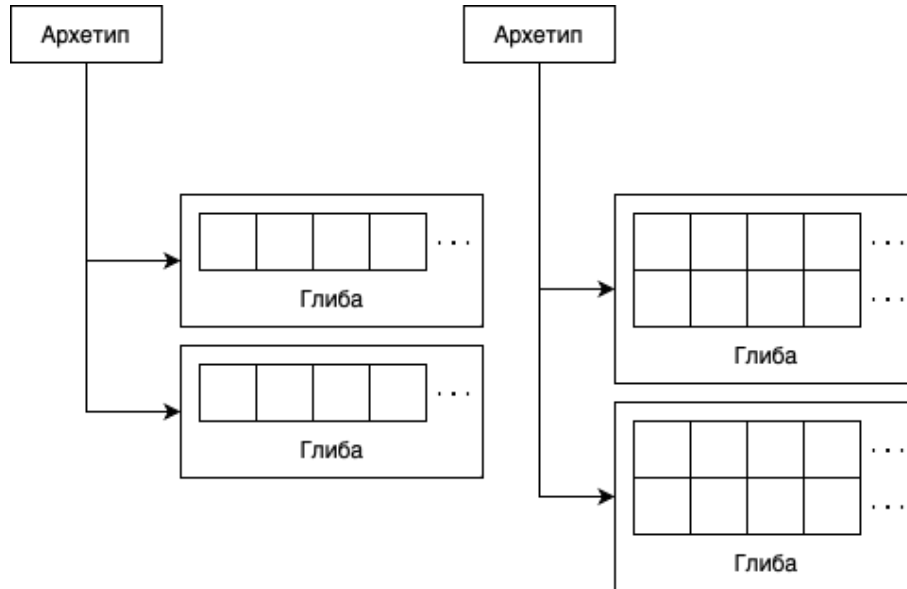


Рис. 1.2: Глиби розміром з більш ніж один масив.

- Одним із них буде спрощення симуляції до двовимірної — таким чином доволі легко емпірично перевірити правильність симуляції.
- Частинки рухатимуться лише поступально — відсутність обертального руху значно спростить тестування, проте його легко додати за допомогою двох додаткових компонент та однієї системи.
- Всі частинки володіють однаковою масою - приймається до уваги лише на останньому кроці дослідження - власне симуляції.

### 1.3 Пружне зіткнення

Розглянемо зіткнення таких двох частинок (див. рис. 1.3). За зіткнення вважатимемо стан пари, в якому відстань між центрами є меншою ніж сума радіусів учасників. Для корекції такої похибки симуляції, кожного

з учасників переміщаємо на половину розміру перетину в напрямку від точки зіткнення.

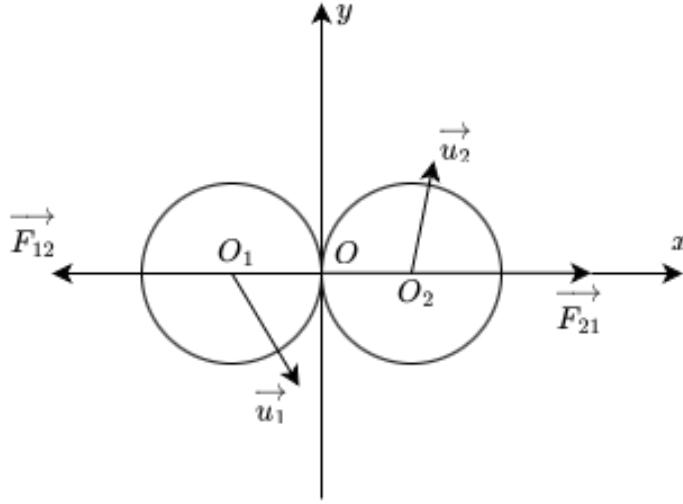


Рис. 1.3: Пружний удар.

Нехай  $\vec{n}$  — вектор напрямку між центрами частинок,  $m_1, m_2$  — маси частинок, а  $\vec{u}_1, \vec{u}_2$  та  $\vec{u}'_1, \vec{u}'_2$  — швидкості частинок до і після зіткнення відповідно. Тоді закони збереження імпульсу та енергії матимуть вигляд:

$$m_1 \vec{u}'_1 + m_2 \vec{u}'_2 = m_1 \vec{u}_1 + m_2 \vec{u}_2 \quad (1.1)$$

$$\frac{m_1 u_1'^2}{2} + \frac{m_2 u_2'^2}{2} = \frac{m_1 u_1^2}{2} + \frac{m_2 u_2^2}{2} \quad (1.2)$$

Оскільки немає сил що діють в напрямку дотичному до поверхонь частинок, тангенціальні складові їхніх швидкостей залишаються незмінними:

$$\begin{aligned} \langle u'_1 | t \rangle &= \langle u_1 | t \rangle \\ \langle u'_2 | t \rangle &= \langle u_2 | t \rangle \end{aligned} \quad (1.3)$$

Прийнявши це до уваги, рівняння 1.1 та 1.2 набувають вигляду:

$$\begin{aligned} m_1 \langle u'_1 | n \rangle + m_2 \langle u'_2 | n \rangle &= m_1 \langle u_1 | n \rangle + m_2 \langle u_2 | n \rangle \\ m_1 \langle u'_1 | n \rangle^2 + m_2 \langle u'_2 | n \rangle^2 &= m_1 \langle u_1 | n \rangle^2 + m_2 \langle u_2 | n \rangle^2 \end{aligned} \quad (1.4)$$

Провівши ряд спрощень:

$$m_1 \langle u'_1 | n \rangle^2 - m_1 \langle u_1 | n \rangle^2 = m_2 \langle u_2 | n \rangle^2 - m_2 \langle u'_2 | n \rangle^2 \quad (1.5)$$

$$\Rightarrow \langle u'_1 | n \rangle + \langle u_1 | n \rangle = \langle u_2 | n \rangle + \langle u'_2 | n \rangle \quad (1.6)$$

Підставивши дану тотожність в закон збереження імпульсу отримуємо:

$$\langle u'_1 | n \rangle = \frac{(m_1 - m_2) \langle u_1 | n \rangle + 2m_2 \langle u_2 | n \rangle}{m_1 + m_2} = \langle u_1 | n \rangle - \frac{2m_2 \langle u_1 - u_2 | n \rangle}{m_1 + m_2} \quad (1.7)$$

З чого і випливає рівняння для кінцевої швидкості першої частинки:

$$u'_1 = \langle u'_1 | n \rangle n + \langle u'_1 | t \rangle t = u_1 - \frac{2m_2}{m_1 + m_2} \frac{\langle u_1 - u_2 | x_1 - x_2 \rangle}{\|x_1 - x_2\|^2} (x_1 - x_2) \quad (1.8)$$

Швидкість другої частинки визначається симетрично:

$$u'_2 = \langle u'_2 | n \rangle n + \langle u'_2 | t \rangle t = u_2 - \frac{2m_1}{m_1 + m_2} \frac{\langle u_2 - u_1 | x_2 - x_1 \rangle}{\|x_2 - x_1\|^2} (x_2 - x_1) \quad (1.9)$$

## 1.4 Довжина вільного пробігу

Щоб обчислювальна складність симуляції зростала однорідно, необхідно змінювати розмір частинок таким чином, щоб довжина вільного пробігу залишалась сталою. Нехай середня швидкість руху частинок в газі буде рівна  $u$ , а  $n$  - концентрація частинок (середня кількість частинок на одиницю простору). Тоді простір, в якому можуть перебувати інші частинку щоб зіткнутись з пробною (див. рис. 1.4) за час  $\Delta t$  визначатиметься як площа прямокутника шириною  $4r$  та довжиною  $u\Delta t$  ( $ABCD$  на рис. 1.4):

$$S = 2rv\Delta t \quad (1.10)$$

таким чином, середня кількість зіткнень через які пройде ця частинка за одиницю часу обчислюватиметься як

$$z = \frac{Sn}{\Delta t} = 4rvn \quad (1.11)$$

Час вільного пробігу  $\tau$  є величиною оберненою до середньої кількості зіткнень, таким чином, відстань вільного пробігу буде рівна

$$\lambda = u\tau = \frac{u}{z} = \frac{u}{4rvn} \quad (1.12)$$

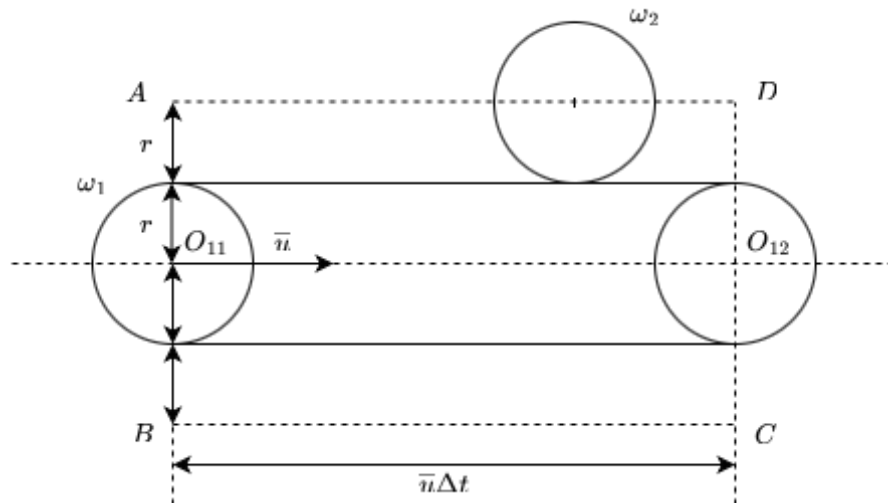


Рис. 1.4: Вільний пробіг.

## Розділ 2

# Експерименти

### 2.1 Формулювання задачі

Проведемо експеримент над описаною задачею згідно постулатів СКС, там під стандартним шаблоном ООП. Для цього проведемо мо ряд симуляцій, по  $10^3$  кадрів кожна, де модуль початкової швидкості частинок сталий, її напрямок випадковий, розмір визначений згідно вищезгаданих принципів, а кількість зростає геометрично, з кроком  $\beta = 1.1$ , в межах  $n \in [1000, 10000]$ .

### 2.2 Результати

Для початку, розглянемо профіль центральної обчислювальної одиниці впродовж симуляції для обох моделей для довільного експерименту (див. рис. 2.1 та 2.2). Розглянемо вершину графу "Next". Помітимо, що СКС зайняв на **16%** менше процесорного часу на обчислення всіх кадрів.

На жаль, профіль купи не несе жодного змісту, оскільки пам'ять яку компілятор вважає за часто використовувану переноситься у стек. Проте у нас немає жодних підстав вважати, що будь-який з підходів споживає більше ресурсів пам'яті ніж інший. Натомість, важливо зазначити, що СКС виділяє всю необхідну йому пам'ять на кроці ініціалізації, в той час як ООП здатен алокувати та задовольняти її і процесі симуляції.

Розглянемо залежність середнього часу розрахунку одного кадру симуляції в залежності від кількості симульованих об'єктів (див. рис. 2.3). Помітимо, що час затрачений СКС є значно менший ніж час затрачений ООП.

File: turbulence  
 Type: cpu  
 Time: May 23, 2023 at 8:48pm (EEST)  
 Duration: 22.50s, Total samples = 21.95s (97.54%)  
 Showing nodes accounting for 21.73s, 99.00% of 21.95s total  
 Dropped 34 nodes (cum <= 0.11s)  
 See <https://git.io/JfYMW> for how to read the graph

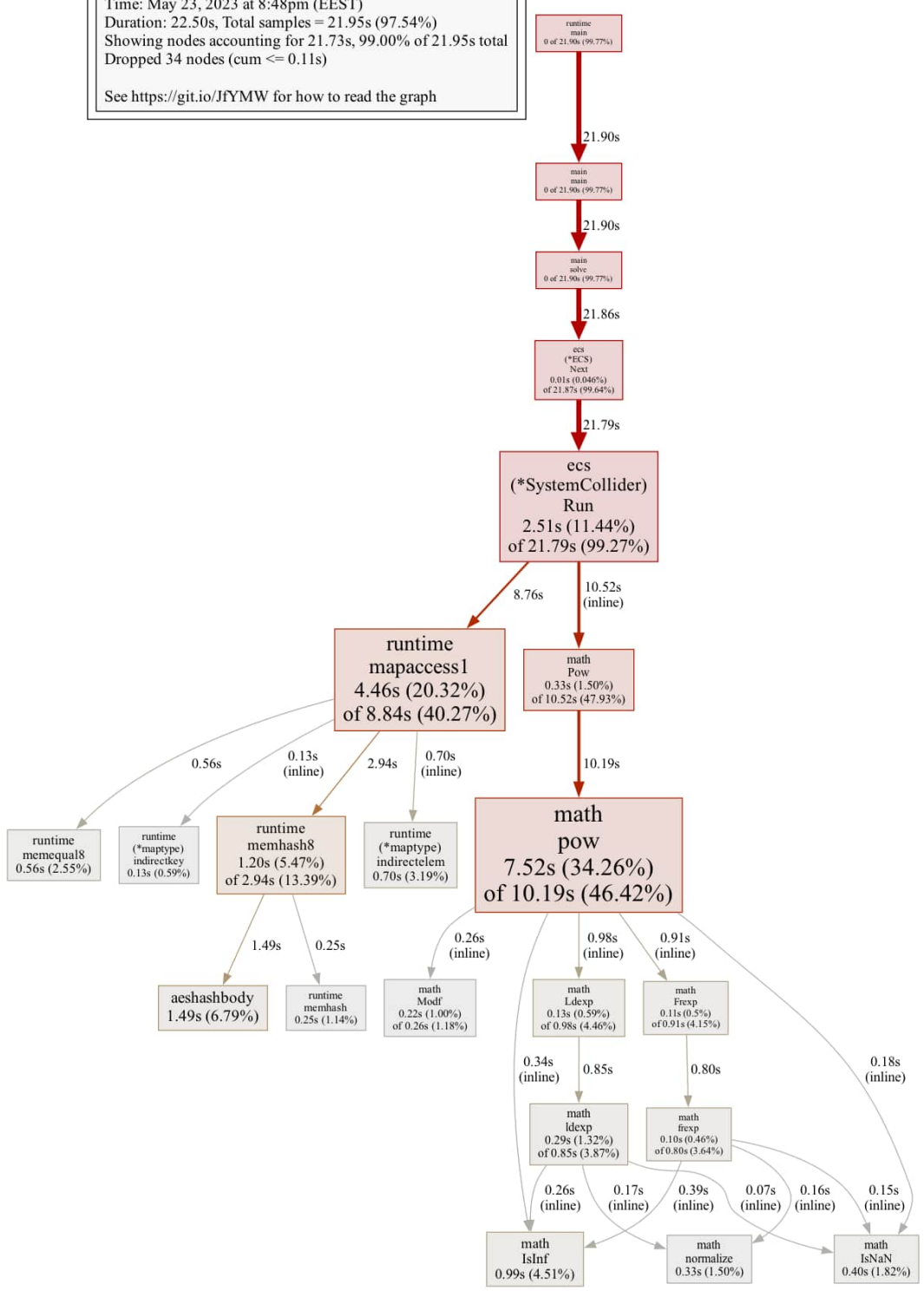


Рис. 2.1: Профіль ЦОУ (СКС).

File: turbulence  
 Type: cpu  
 Time: May 23, 2023 at 10:31pm (EEST)  
 Duration: 26.05s, Total samples = 25.19s (96.71%)  
 Showing nodes accounting for 25.15s, 99.84% of 25.19s total  
 Dropped 26 nodes (cum <= 0.13s)  
 See <https://git.io/JfYMW> for how to read the graph

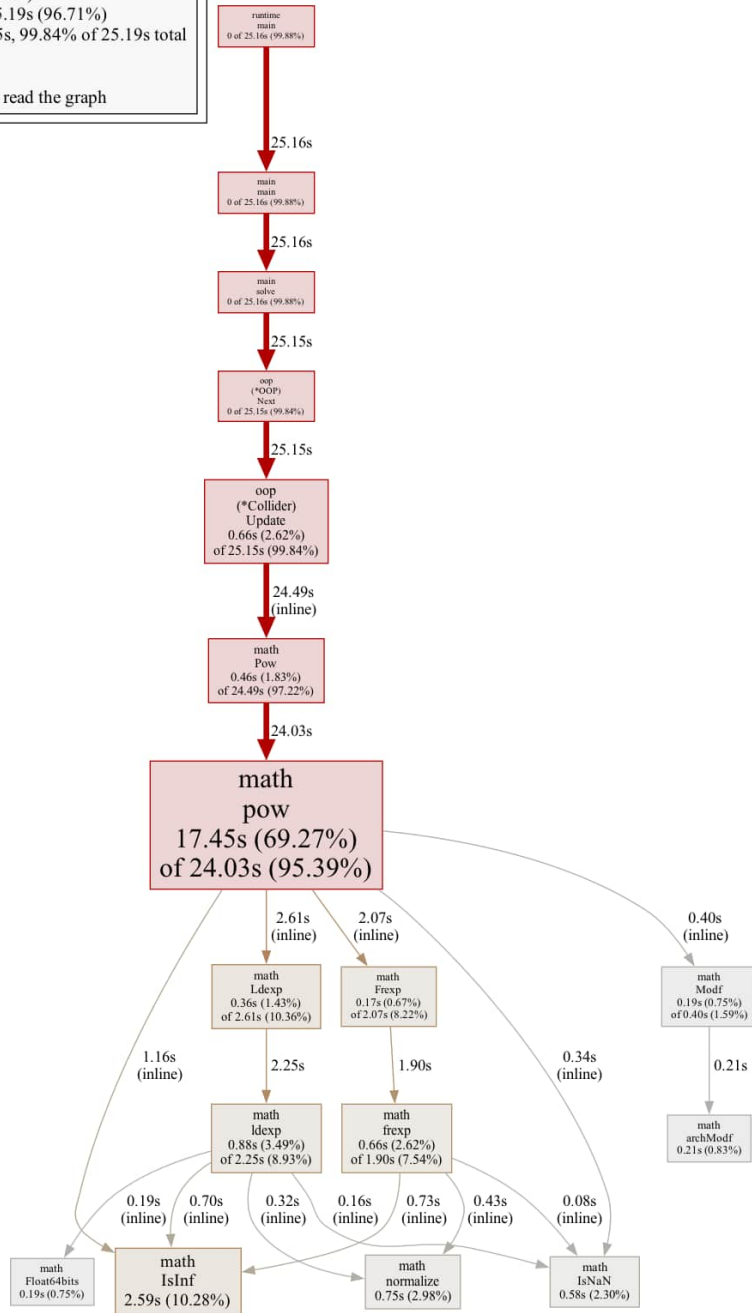


Рис. 2.2: Профіль ЦОО (ООП).

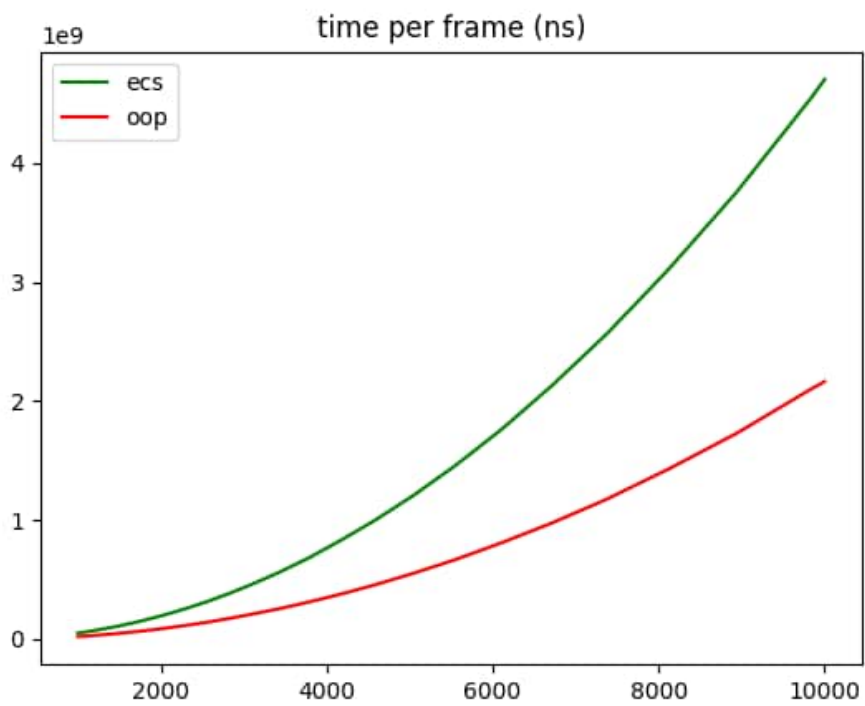


Рис. 2.3: Середній час розрахунку одного кадру.

## Розділ 3

# Висновки

Використання СКС беззаперечно вносить свої переваги у роботу програмного забезпечення, проте його особливості обмежують можливості його прикладного застосування. Великим недоліком даного підходу є збільшення ресурсів витрачених на розробку продукту, проте роздуми щодо вигоди використання цього методу в корпоративних цілях є поза межами цієї роботи.

## Розділ 4

### Джерела

1. A. Tanenbaum - Distributed Systems: Principles and Paradigms / Andrew S. Tanenbaum, Maarten Van Steen - Pearson, 2016. – 21-513 с.
2. D. A. Patterson - Computer Organization and Design: The Hardware/Software Interface / David A. Patterson, John L. Hennessy - Morgan Kaufmann, 2017. – 97-311 с.
3. J. L. Hennessy - Computer Architecture: A Quantitative Approach / John L. Hennessy, David A. Patterson - Morgan Kaufmann, 2017. – 59-220 с.
4. W. Stallings - Operating Systems: Internals and Design Principles / William Stallings - Pearson, 2017. – 300-601 с.
5. J. F. Kurose - Computer Networking: A Top-Down Approach / James F. Kurose, Keith W. Ross - Pearson, 2017. – 49-401 с.
6. M. L. Liu - Real-Time Systems / Jane W. S. Liu - Pearson, 2017. –

38-315 c.

7. R. H. Katz - Contemporary Logic Design / Randy H. Katz, Gaetano Borriello - Pearson, 2019. 10-95 c.
8. M. V. Cai - Advanced Computer Architecture: Parallelism, Scalability, Programmability / Mike V. Cai - CRC Press, 2019. 15-85 c.