УДК 004.42

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СКІНЧЕННО-ЕЛЕМЕНТНОЇ ДИСКРЕТИЗАЦІЇ 2D ОБЛАСТЕЙ З ВИКОРИСТАННЯМ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ CUDA

Я.І. Соколовський[1], А.В. Нечепуренко[2], О.П. Герасимчук[3]

Здійснено програмну реалізацію скінченно-елементної дискретизації двовимірної області за допомогою алгоритму "прямої побудови". Програмне забезпечення для автоматизації скінченно-елементної дискретизації розроблено на платформі .NET Framework мовою програмування C# з використанням технологій паралельних обчислень CUDA. Проектування програмної системи здійснено за допомогою UML діаграм. Створений інтерфейс користувача дає змогу задавати параметри тріангуляції та розбивати параметри дискретизації у заданих областях.

Ключові слова: об'єктно-орієнтована модель, тріангуляція, скінченно-елементна дискретизація, k-d дерево, CUDA.

Вступ. На сьогодні для отримання чисельного розв'язку задач тепло- та вологоперенесення у капілярно-пористих матеріалах широкого застосування набув метод скінченних елементів з уточненням сіткової апроксимації. Одним з основних етапів скінченно-елементного аналізу є дискретизація розрахункової області, від якої напряму залежить точність розв'язків задачі. Окрім цього, розбиття області на скінченні елементи – доволі трудомісткий процес. Тому автоматизація процесу дискретизації є актуальним завданням сучасних програмних комплексів для скінченно-елементних розрахунків. Пошук ефективного вирішення такого завдання зумовлює впровадження технологій паралельних обчислень у процес дискретизації досліджуваної області. У цьому напрямі перспективним є застосування архітектури паралельних обчислень CUDA (Compute Unified Device Architecture). Ці технології дають змогу істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів, а саме, дає змогу організовувати доступ до набору інструкцій графічного прискорювача і керувати його пам'яттю. Застосування цієї архітектури для розроблення програмного забезпечення скінченно-елементного аналізу дає змогу досягти високої продуктивності процесу дискретизації області, що, своєю чергою, позитивно впливає на точність отриманих скінченно-елементних розрахунків.

1. Аналіз наявних досліджень

Дискретизація розрахункової області є невід'ємною частиною скінченно-елементних розрахунків. Від процесу побудови скінченно-елементної сітки та її якості залежить як затрата ресурсів і трудомісткість майбутньої програми, так і певні похибки в результатах розрахунків. Проте існує невелика кількість програмного забезпечення з відкритим програмним кодом для проведення тріангуляції на основі критерію Делоне з використанням технологій паралельних обчислень. У деяких програмних комплексах не створені візуальні інтерфейси або їхні програмні коди написані на мовах програмування, що обмежує інваріантність програм, використання готових компонент та можливу взаємодію з іншими програмними продуктами. Таке програмне забезпечення охоплює велику кількість трудомістких процесів, що навіть для обчислювальних можливостей сучасних процесорів значно обмежує роботу програми при дискретизації досліджуваної області та згущення вузлової сітки. Розроблення програмних комплексів скінченно-елементної дискретизації з використанням технологій паралельних обчислень дасть змогу усунути зазначені вище недоліки.

Є кілька видів алгоритмів побудови тріангуляції **[**7-12**]**. В основі ітераційних алгоритмів закладено ідею поступового додавання вузлів в частково побудовану тріангуляцію Делоне. Складність цього підходу полягає у трудомісткості пошуку трикутника, в який на кожному наступному кроці додається вузол. Побудова нових трикутників, а також перебудова існуючої структури тріангуляції, отриманої внаслідок незадовільних перевірок пар сусідніх трикутників на виконання умови Делоне, є доволі складними завданнями. Трудомісткість ітераційних алгоритмів в середньому становить $O(N^{3/2})$.

Іншим підходом до дискретизації області множиною трикутників є алгоритми тріангуляції злиттям **[**10, 11**]**. Вони передбачають розбиття вихідної множини точок на кілька підмножин. На наступному кроці проводиться тріангуляція цих підмножин, після чого відбувається процес злиття кількох тріангуляцій в одну. Трудомісткість таких алгоритмів у середньому сягає $O(N \log N)$.

Описані вище алгоритми мають одну особливість, а саме – перебудову структури тріангуляції на певних етапах. При побудові тріангуляції ітеративними алгоритмами або алгоритмами злиття, після побудови кожного нового трикутника повинна проводитись перевірка на виконання умови Делоне для цього трикутника. Окрім цього, доводиться проводити описану вище перевірку ще для трьох сусідніх трикутників. Якщо перевірка виявила не відповідність тріангуляції критерію Делоне, то потрібно проводити перебудову трикутників і нову серію перевірок на відповідність. На практиці, доволі багато часу займають саме перевірки виконання умови Делоне та перебудови трикутників.

Для спрощення роботи ітераційних алгоритмів та алгоритмів тріангуляції злиттям можна на першому етапі будувати тріангуляцію, ігноруючи перевірку виконання умови Делоне. На другому етапі проводиться перевірка тріангуляції на виконання умови Делоне та за потреби проводяться перебудови. Такий підхід реалізовано у двопрохідних алгоритмах тріангуляції. Трудомісткість двопрохідних алгоритмів в середньому становить $O(N \log N)$.

Ще однією групою алгоритмів тріангуляції є алгоритми прямої побудови **[**11, 12**]**. Їх суть полягає в тому, щоб відразу будувати тільки такі трикутники, які відповідають критерію Делоне і не потребують перебудови. Протягом тривалого часу цей підхід рідко реалізовувався на практиці у програмних продуктах для скінченно-елементних розрахунків, оскільки є доволі трудомістким та становить $O(N^2)$. Проте розвиток технологій паралельних обчислень дав новий поштовх у практичному застосуванні алгоритмів прямої побудови.

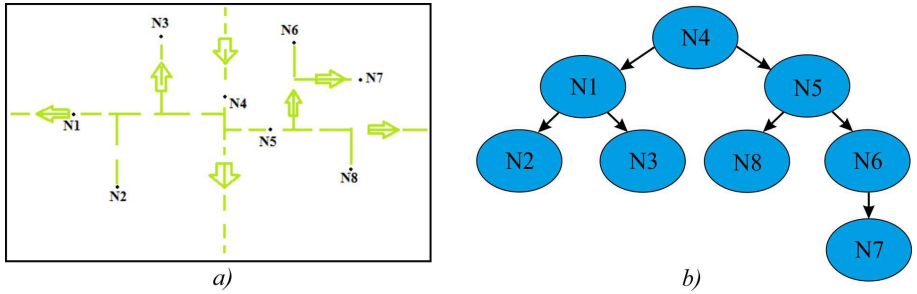
У роботі розглянуто паралельну реалізацію алгоритму "прямої побудови" тріангуляції Делоне у контексті реалізації методу скінченних елементів за допомогою архітектури паралельних обчислень CUDA, що дає змогу істотно збільшити обчислювальну продуктивність за рахунок використання потужності гра-

[1] проф. Я.І. Соколовський, д-р техн. наук – НЛТУ України, м. Львів;
[2] асист. А.В. Нечепуренко – НЛТУ України, м. Львів;
[3] асист. О.П. Герасимчук – НЛТУ України, м. Львів.

фічного процесора (GPU). Платформа паралельних обчислень CUDA забезпечує набір розширень для мов програмування C і C ++, що дають змогу висловлювати як паралелізм даних, так і паралелізм завдань на рівні дрібних і великих структурних одиниць[4-6].

2. Алгоритмічні аспекти методу "Прямої побудови" тріангуляції областей

Досліджувану двовимірну прямокутну область визначено декартовим добутком $[x_1, x_2]\times[y_1, y_2]$ де $x\in[x_1, x_2]$ і $y\in[y_1, y_2]$. Для розміщення N_i вузлів з координатами (x_i, y_i) на площині використовуємо генератор випадкових чисел. На множині згенерованих вузлів будується структура k-d дерева [12]. Суть побудови дерева полягає у розділенні площини на дві півплощини прямою, паралельною одній із координатних осей, наприклад осі OY. Надалі, кожна з цих півплощин може розділятися повторно прямою, паралельною іншій координатній осі. Ці дії повторюються, змінюючи на кожному кроці напрям лінії розділення. Лінії, що розділяють площину, вибираються за принципом простої дихотомії, тобто, керуючись отриманням приблизно рівного числа вузлів по кожну сторону від лінії розділення (рис. 1). Такий принцип є основою ідеї методу багатовимірного двійкового дерева [12].



a) *b)*

Рис. 1. Ілюстрація методу пошуку з допомогою двовимірного двійкового дерева: *a) схема розділення площини на підплощини; b) k-d дерево, сформоване з вузлів площини, зображених на a)*

Отримавши k-d дерево, застосовується метод "Прямої побудови" для проведення тріангуляції на досліджуваній області. Для цього на першому кроці вибирається базова лінія AB з координатами $(X_1, Y_1; X_2, Y_2)$ (рис. 2), яка є стороною початкового трикутника V_1 з вершинами $(X_{11}, Y_{11}; X_{21}, Y_{21}; X_{31}, Y_{31})$. Наступним кроком є побудова уявного квадрата K, діагональ якого є стороною AB трикутника V_1. Далі відбираються вузли $C(x_{0k}, y_{0k})$, які потрапили в побудований квадрат. Якщо у квадрат не потрапило жодного вузла, то розміри квадрата збільшуються доти, поки хоча б один вузол не потрапить в нього. При потраплянні вузла $C(x_{0k}, y_{0k})$ у квадрат K, будується уявний трикутник W_k з вершинами $(X_{0k}, Y_{0k}; X_1, Y_1; X_2, Y_2)$, однією стороною якого буде базова лінія AB, а інші сторони будуються зв'язуванням кінців цієї лінії з вибраним вузлом C. Для кожної нової побудови уявного трикутника W_k проводиться перевірка виконання умо-

ви Делоне і відкидаються ті трикутники, для яких умова не справджується. Серед таких W_k, які пройшли перевірку, вибирається трикутник з мінімальним радіусом R, після чого ці трикутники вносяться в структуру попередньо збудованої тріангуляції. При побудові кожного наступного трикутника V_k, з вершинами $(X_{1k}, Y_{1k}; X_{2k}, Y_{2k}; X_{3k}, Y_{3k})$ базова лінія змінюється та повторно проводиться перевірка умови Делоне для вузлів, які потрапили у новий квадрат, побудований відносно нової базової лінії. Вибір сторони трикутника, яка буде виступати базовою лінією, проводиться за допомогою пошуку вузлів в k-d дереві. З k-d дерева вибирається два вузли, для яких сума відповідних координат буде максимальною $\{X_{\max}, Y_{\max}\} = \max\{X_{1k} + X_{2k}; Y_{1k} + Y_{2k}\}$. Між цими вузлами будується лінія, яка буде базовою. Вибір вузлів буде здійснюватись доти, доки сторона нового побудованого трикутника не буде належати верхній границі досліджуваної прямокутної області. Після чого умова вибору вузлів базової лінії з k-d дерева змінюється на протилежну.

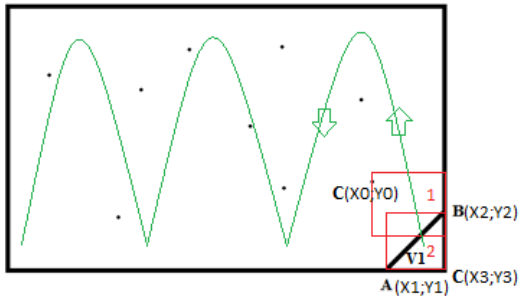


Рис. 2. Схема алгоритму "Прямої побудови"

Тріангуляцією Делоне є така тріангуляція, в якій в радіус описаного кола навколо будь-якого елемента не потрапляє жоден інший вузол цієї тріангуляції [11]. Основна ідея алгоритму перевірки умови Делоне полягає у знаходженні центра (*a; b*) і радіуса R описаного кола:

$$\begin{aligned} AB &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ BC &= \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}. \\ AC &= \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \end{aligned} \quad (1)$$

Знайшовши сторони трикутників розбиття та скориставшись формулою Герона, обчислимо радіус описаного кола:

$$\begin{aligned} P &= \frac{AB + BC + AC}{2} \\ S &= \sqrt{pbc \cdot (p - AB)(p - BC)(p - AC)}. \\ R &= \frac{AB \cdot BC \cdot AC}{4S} \end{aligned} \quad (2)$$

Отримавши радіус, підставляємо його в рівняння кола:

$$\begin{cases}(x_1-a)^2+(y_1-b)^2=R^2\\(x_2-a)^2+(y_2-b)^2=R^2.\\(x_3-a)^2+(y_3-b)^2=R^2\end{cases} \quad (3)$$

Знаходимо координати центра кола:

$$a=b\frac{y_2-y_1}{x_1-x_2}+\frac{y_1^2-y_2^2}{2(x_1-x_2)}+\frac{x_1+x_2}{2};$$

$$b=\frac{2\left(\left(\frac{y_2-y_1}{x_1-x_2}\right)\left(x_3-\left(\frac{y_1^2-y_2^2}{2(x_1-x_2)}+\frac{x_1+x_2}{2}\right)\right)+y_1\right)+\sqrt{4D}}{2\left(\left(\frac{y_2-y_1}{x_1-x_2}\right)+1\right)};$$

$$D=4\left(\frac{y_2-y_1}{x_1-x_2}\left(x_3-\left(\frac{y_1^2-y_2^2}{2(x_1-x_2)}+\frac{x_1+x_2}{2}\right)\right)\right)^2+y_1-$$

$$-4\left(\left(\frac{y_2-y_1}{x_1-x_2}\right)^2+1\right)\left(\left(x_3-\left(\frac{y_1^2-y_2^2}{2(x_1-x_2)}+\frac{x_1+x_2}{2}\right)\right)^2+y_1^2-R^2\right). \quad (4)$$

Тоді умова Делоне для трикутника з вершинами $(X_1,Y_1),(X_2,Y_2),(X_3,Y_3)$ буде виконуватись тоді, коли для будь-якого іншого вузла (X_0,Y_0) тріангуляції буде справджуватись нерівність

$$(X_0-a)^2+(Y_0-b)^2\geq R^2. \quad (5)$$

Якщо такі вузли існують, то з них вибирається новий $C(x_3,y_3)$. Ця процедура виконується доти, доки будуть вузли, що задовольняють умову (5).

3. Аспекти програмної реалізації розпаралелювання алгоритму тріангуляції "прямої побудови"

На першому етапі програмної реалізації тріангуляції на основі алгоритму "прямої побудови" здійснено проектування майбутньої системи за допомогою UML діаграм. Діаграма прецедентів дає змогу подати функціональні можливості проектованого програмного забезпечення. Користувач може здійснювати генерування вузлів у досліджуваній двовимірній прямокутній області, а також відображати результати тріангуляції (рис. 3).

Структуру моделі системи в термінології класів об'єктно-орієнтованого програмування представлено на діаграмі класів (рис. 4). Клас "Engeen" є основним компонентом програми. У ньому містяться методи для формування списку вузлів (CreatePoints), перевірки виконання умови Делоне (CheckDelone), побудови трикутників та їх сторін (CreateTriangle, RegisterSide). Метод для формування вузлів "CreatePoints" зв'язаний з класом "KDTree" та "KDNode". За допомогою класу "KDNode" формується k-d дерево. Своєю чергою, у класі

"KDTree" використовуються методи "Insert", "Delete", "Search", "Nearest", які максимально розширюють спектр роботи з деревом. У класі "Side" містяться конструктор побудови сторони трикутника та методи, які потрібні на проміжних розрахунках побудови тріангуляції. Одним із таких методів є "CrossSide", який перевіряє чи перетинає побудований трикутник сусідні. Клас "Triangle" аналогічно класу "Side", містить конструктор побудови трикутника та методи для визначення площі трикутника (GetSquare), центр описаного навколо трикутника кола (CentreCircle), та чи попала точка в середину описаного кола (IsInCircle). Класи "Triangle" та "Side" зв'язані з класом "Point2D" та містять об'єкти цього класу.

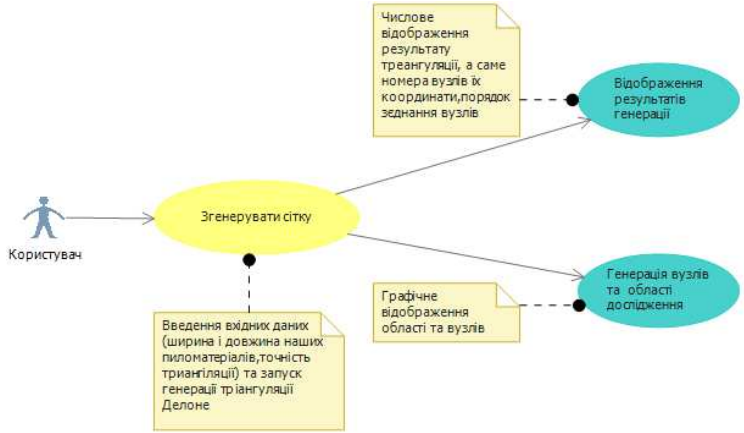


Рис. 3. Діаграма варіантів використання

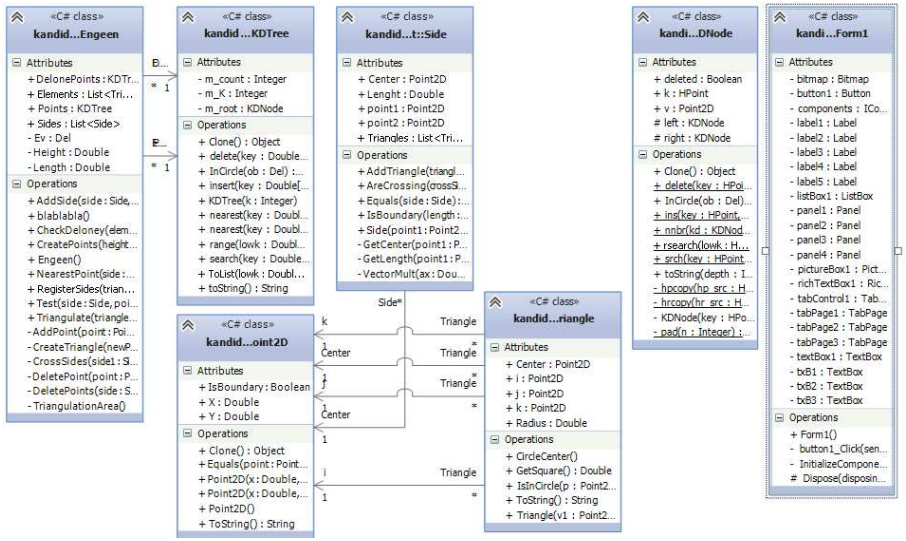


Рис. 4. Діаграма класів

У зв'язку з тим, що реалізація алгоритму "прямої побудови" є доволі трудомісткою, використано технології паралельних обчислень CUDA для багаторазових перевірок виконання умови Делоне. Кількість паралельних потоків, використаних у програмній реалізації, дорівнює кількості операцій, яку вимагає алгоритм перевірки умови Делоне (обчислення центра і радіуса описаного навколо трикутника кола). Відповідно, кожен потік блоку відповідає за певну операцію, необхідну для обчислення критерію Делоне трикутника з вершинами $W_k(X_{0k}, Y_{0k}; X_1, Y_1; X_2, Y_2)$. Зокрема, *block*(0; 0) відповідає за обчислення (рис. 5) критерію Делоне для трикутника $W_0(X_0, Y_0; X_1, Y_1; X_2, Y_2)$ і т. д.

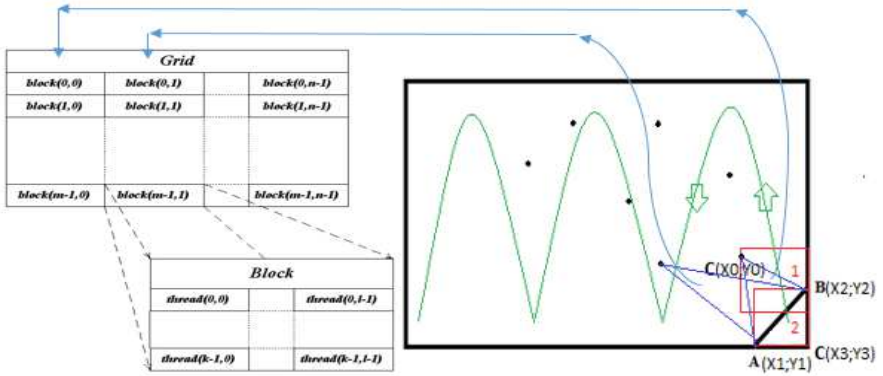


Рис. 5. Схема потоків паралельного виконання операцій перевірки умови Делоне

Основною мовою програмування в архітектурі CUDA є C. Але для програмування у середовищі Visual Studio мовою C# з використанням GPGPU використовується CUDAfy[11]. Модуль, який виконує перетворення вхідних потоків у вихідні, є ядром (kernel). Окремі ядра можуть сполучатися між собою, що призводить до утворення досить складних схем оброблення потоків вхідних даних. Виклик модуля проводиться з тіла програми (метод findrightP), а його оброблення відбувається в методі kernel, де і визначається кількість потоків, на які розбивається завдання. У цьому випадку потоків буде така ж кількість, як і вузлів, на яких буде проводитись тріангуляція.

Фрагменти виконання тріангуляції області за допомогою архітектури паралельних обчислень наведено в такому коді:

```
[Cudafy]
public static void kernel (GThread thread, List<Point> AnalizeP, List<Point> AnalizeP1, double AB, double b)
{ int tid = thread.blockIdx.x;
for (int i = 0; i < AnalizeP.Count; i++)
{
if (tid < AnalizeP.Count && AnalizeP[tid].X > b – AB && AnalizeP[tid].Y < AB)
{ AnalizeP1.Add(AnalizeP[tid]); }}}
public static List<Point> findrightP(double AB, double b, List<Point> AnalizeP)
{ CudafyModule km = CudafyTranslator.Cudafy();
GPGPU gpu = CudafyHost.GetDevice(CudafyModes.Target, CudafyModes.DeviceId);
gpu.LoadModule(km);
List<Point> AnalizeP1 = new List<Point>();
```



```
// launch add on AnalizeP.Count threads
gpu.Launch(AnalizeP.Count, 1).kernel(AnalizeP, AnalizeP1, AB, b);
return AnalizeP1; }
```

На рис. 6 представлено інтерфейс користувача головного вікна програми. Перед тим як здійснити тріангуляцію, потрібно задати геометричні розміри досліджуваної області, кількість вузлів та вид їх зосередження.

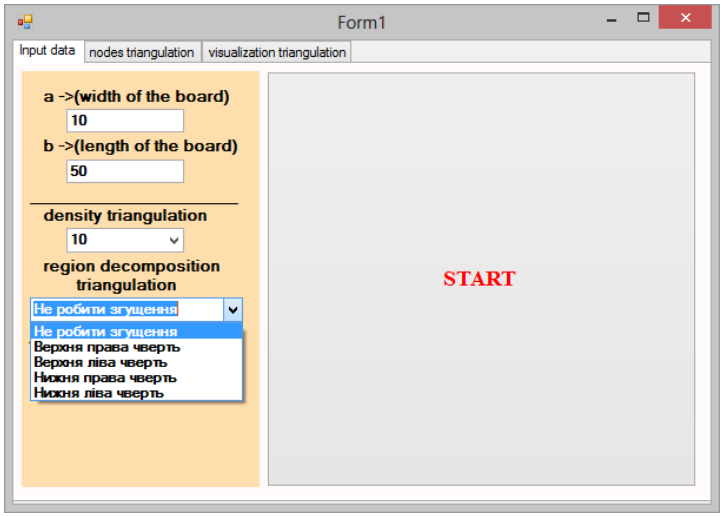


Рис. 6. Інтерфейс користувача програми

Перед тим як провести тріангуляцію, здійснюється аналіз розташування вузлів та послідовно обирається оптимальний вузол для побудови нового трикутника. Дані про такі вузли і послідовну побудову трикутників проілюстровано на рис. 7.

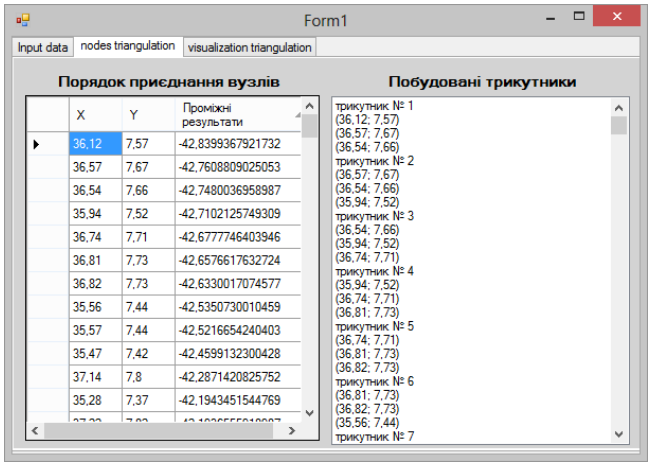


Рис. 7. Параметри побудованої тріангуляції

На основі геометричних параметрів тріангуляції будується матриця у контексті застосування методу скінченних елементів із значень функцій форми симплекс-елементів. Ця матриця буде використовуватися для побудови системи рівнянь, потрібних для розв'язку задачі нестаціонарного тепломасоперенесення у двовимірній області.

Висновки. Розроблено прикладне програмне забезпечення для автоматизації процесу тріангуляції двовимірних областей з використанням архітектури паралельних обчислень CUDA. Створений графічний інтерфейс дає змогу задати геометричні параметри досліджуваної області, а також отримувати кількісні характеристики тріангуляції для оцінки ефективності роботи алгоритмів.

Паралельна реалізація алгоритму тріангуляції "прямої побудови" дає змогу зменшити час, потрібний для проведення дискретизації досліджуваної області, та зменшити затрати пам'яті, завдяки використанню ресурсів графічного процесора. Це програмне забезпечення буде інтегровано до програмного комплексу для скінченно-елементного розрахунку тепломасоперенесення у капілярно-пористих матеріалах.

Література

1. Ильин В.П. Методы и технологии конечных элементов / В.П. Ильин. – Новосибирск. – 2007. – С. 370-376.
2. [Електронний ресурс]. – Доступний з http://kniga.scienceontheweb.net/variant-ispolzovaniya
3. [Електронний ресурс]. – Доступний з http://www. uk.wikipedia.org.
4. [Електронний ресурс]. – Доступний з http://www.nvidia.ru/object/cuda-parallel-computing-ru.html
5. [Електронний ресурс]. – Доступний з http://www.nvidia.ru/object/cuda-openacc-online-course-ru.html
6. [Електронний ресурс]. – Доступний з http://vk.com/nvidiacuda.
7. Яненко Н.Н. О вариационном методе построения сеток // Численные методы механики сплошной среды // Яненко Н.Н., Данаев Н.Т., Лисейкин В.Д. – Новосибирск. – 1987. – Т. 8, № 4. – С. 157-163.
8. Копысов С.П. Сравнение r- и h-версий МКЭ на примере задач теории упругости / С.П. Копысов, А.К. Новиков // Актуальные проблемы прикладной математики и механики : матер. Междунар. науч.-техн. конф. 3-7 фев., 2003 г. – Екатеринбург, 2003. – С. 44-45.
9. Новиков А.К. Параллельная h-версия МКЭ в решении двухмерных задач теории упругости / А.К. Новиков. – Ижевск, 2004. – С. 6-18.
10. Скворцов А.В. Обзор алгоритмов триангуляции Делоне / А.В. Скворцов // Вычислительные методы и программирование : сб. науч. тр. – 2002. – Т. 3. – С. 14-39.
11. Скворцов А.В. Триангуляция Делоне и её применение / А.В. Скворцов. – Томск : Изд-во Томского ун-та, 2002. – С. 128.
12. Скворцов А.В. Эффективнее алгоритмы построения триангуляции Делоне / А.В. Скворцов, Ю.Л. Костюк // Геоинформатика. Теория и практика : сб. науч. тр. – Томкс : Изд-во Томского ун-та. – 1998. – Вып. 1. – С. 22-47.
13. Яненко Н.Н. О вариационном методе построения сеток / Н.Н. Яненко, Н.Т. Данаев, В.Д. Лисейкин // Численные методы механики сплошной среды : сб. науч. тр. – Новосибирск : Изд-во "Наука". – 1987. – Т. 8, № 4. – С. 157-163.
14. DSP Hybrid. CUDAfy user manual / Hybrid DSP // CUDAfy user manual. – USA. – 2015. – С. 45-48.
15. Препарата Ф. Вычеслительная геометрия / Ф. Препарата, М. Шеймос. – USA. – 1989. – С. 94-97.
16. Соколовський Я.І. Програмний комплекс автоматизації скінченно-елементної дискретизації двовимірних областей / Я.І. Соколовський, О.П. Сикала // Фізико-математичне моделювання та інформаційні технології. – Львів : Вид-во Центру математичного моделювання ін-ту прикладних проблем механіки і математики ім. Я.С. Підстригача НАН України. – 2014. – Вип. 19. – С. 176-188.

Надійшла до редакції 15.09.2016 р.

Соколовский Я.И., Нечепуренко А.В., Герасимчук О.П. **Программное обеспечение конечно-элементной дискретизации 2D областей с использованием параллельных вычислений CUDA**

Осуществлена программная реализация конечно-элементной дискретизации двумерной области с помощью алгоритма "прямой построения". Программное обеспечение для автоматизации конечно-элементной дискретизации разработано на платформе .NET Framework на языке программирования C# с использованием технологий параллельных вычислений CUDA. Проектирование программной системы осуществлено с помощью UML диаграмм. Созданный интерфейс пользователя позволяет задавать параметры триангуляции и разбивать параметры дискретизации в заданных областях.

Ключевые слова: объектно-ориентированная модель, триангуляция, конечно-элементная дискретизация, k-d дерево, CUDA.

Sokolovskyy Y.I., Nechepurenko A.V., Gerasymchuk O.P. **Software Finite-Element Sample 2D Domains Using CUDA Parallel Computing**

Software implementation of finite-element two-dimensional sampling area using the direct construction algorithm is made. Software for automation of finite-element sample is developed on the .NET Framework programming language C # using technologies of parallel computing CUDA. Design software system is implemented using UML diagrams. Created user interface allows you to set parameters triangulation and break options sampling in specified areas.

Keywords: object-oriented model, triangulation, finite-element sampling, k-d tree, CUDA.

УДК 621.3

МОДЕЛЮВАННЯ ПРОЦЕСУ ДЛЯ ОБЧИСЛЕННЯ КЛАСТЕРІВ У БАЗАХ ДАНИХ ДЛЯ ТЕХНОЛОГІЧНИХ БІЗНЕС-ПРОЦЕСІВ

Є.Н. Федорчук[1], Ю.Є. Федорчук[2]

Розглянуто задачі та алгоритми пошуку кластерів у базах даних. Розширено класи задач із використанням оптимізаційних критеріїв для оцінки кластера.

Проаналізовано ефективність підходу до обчислення кластерів у базах даних на основі оптимізаційних критеріїв. Підхід ґрунтується на зведенні дискретної задачі оптимізації до неперервної задачі. Така апроксимація забезпечує швидкий пошук кластерів з контрольованою похибкою. Проаналізовано два алгоритми вирішення задачі. Перший – на основі розбиття області обмежень. Другий – на основі ітераційних алгоритмів. Практична реалізація алгоритмів використовує дві групи операцій. Перша група виконує обчислення елементів кластера в області неперервних обмежень. Друга група виконує уточнення елементів кластера в області дискретних обмежень для бази даних. Наведено алгоритм, технологію і результати пошуку кластерів для бази даних.

Ключові слова. кластер, задача кластеризації, оптимізаційний критерій, дискретна задача оптимізації, розбиття області обмежень, база даних, бізнес-процес.

Вступ. У задачах пошуку кластерів у БД, призначених для підтримки технологічних бізнес-процесів, часто використовують умовні поняття для назв кластерів. Це можуть бути: конфігурації комп'ютерів **[**1**]**, набори товарів (продуктовий кошик) **[**2**]**, послуг, банківські портфелі, логістичні транспортні кластери **[**3**]**. Основною характеристикою таких кластерів часто є такі найбільш поширені види оцінок-критеріїв:

[1] доц. Є.Н. Федорчук, канд. техн. наук – НУ " Львівська Політехніка";
[2] аспір. Ю.Є. Федорчук – Львівський торговельно-економічний університет.