УДК 519.174.1

Мартинюк П.М., к.ф.-м.н., доцент, Семенчук Ю.А., ст. 5 курсу ФПМіКІС
(Національний університет водного господарства та природокористування, м. Рівне)

ВУЗЛОВЕ І СКІНЧЕННОЕЛЕМЕНТНЕ ПОКРИТТЯ ДВОВИМІРНИХ ОБЛАСТЕЙ: ДЕЯКІ АЛГОРИТМИ ТА ЇХ ПРОГРАМНА РЕАЛІЗАЦІЯ

Зроблено огляд деяких алгоритмів вузлового покриття та тріангуляції Делоне двовимірних однозв'язних багатокутних областей. Запропоновано алгоритм видалення трикутників у випадку тріангуляції неопуклих областей. Розглянуті алгоритми програмно реалізовано з використанням сучасної технології візуально–подійного програмування на мові C#. Наведено ряд прикладів, які показали ефективність створеного програмного забезпечення.
Ключові слова: вузлова сітка; тріангуляція; безсітковий метод.

Сделан обзор некоторых алгоритмов узлового покрытия и триангуляции Делоне двумерных односвязных многоугольных областей. Предложен алгоритм удаления треугольников в случае триангуляции невыпуклых областей. Рассмотренные алгоритмы программно реализованы с использованием современной технологии объектно-ориентированного программирования на языке C#. Численные эксперименты показали эффективность созданного программного обеспечения.
Ключевые слова: узловая сетка; триангуляция; безсеточный метод.

An overview of some algorithms for node covers of two-dimensional polygonal areas and some algorithms for Delaunay triangulation have been done. Triangle removal algorithm in the case of triangulation of nonconvex domains has been proposed. The algorithms were implemented using modern technology object oriented programming. Results of numerical investigations and their analysis are presented.
Key words: mesh of nodes; triangulation; meshfree method.

На сьогодні серед чисельних методів розв'язування крайових задач математичної фізики все більшої популярності набувають безсіткові методи: метод частинок, метод радіальних базисних функцій, метод найменших квадратів, метод Петрова – Гальоркіна та його модифікації тощо [8]. **Для застосування безсіткових методів область, в якій шукається розв'язок задачі, покривається вузловою сіткою.** Зараз немає єдиного універсального методу вузлації. Кожен з існуючих методів має свої переваги та недоліки. Тому використовують способи вузлації відповідно до геометричної та топологічної форми області, вимог щодо швидкості роботи алгоритму, рівномірності

та густоти покриття, простоти програмної реалізації. В даній статті **розглянуто та програмно реалізовано декілька алгоритмів вузлації.**

Метод скінченних елементів (МСЕ) широко використовується в галузях проектування, будівництва, при розв'язуванні крайових задач математичної фізики, знаходження розв'язків інтегральних рівнянь та ін.[8]. Саме тому дослідження та розробка алгоритмів скінченноелементного покриття областей була і залишається актуальною проблемою прикладної математики. Загалом **задачі вузлації та скінченноелементного розбиття взаємопов'язані** – вирішення однієї з них дозволяє вирішити іншу.

Розглянемо задачу вузлації однозв'язних областей в двовимірному випадку. Нехай маємо деяку скінченну множину точок $P_i = (x_i, y_i), i = \overline{1,n}$, які утворюють однозв'язну многокутну замкнену область Г. Внутрішність та межу даної області потрібно вкрити вузлами. Розглянемо декілька алгоритмів вузлації.

Алгоритм випадкового покриття вузлами на основі генератору псевдовипадкових чисел. Цей алгоритм найчастіше використовується для вирішення вищепоставленої задачі, оскільки є відносно простим в реалізації і не є ресурсозатратним. В його основі лежать наступні кроки: 1) вписання області Г в прямокутник з межами по максимальних і мінімальних значеннях абсцис та ординат точок межі; 2) генерація внутрішніх вузлів за допомогою генератору псевдовипадкових чисел за межами утвореного прямокутника; 3) видалення вузлів, які не належать області Г. У представленій модифікації методу є можливість задавати найменшу відстань згенерованого вузла до інших точок, що дає можливість керувати рівномірністю покриття. Вузли на межі області Г генеруються за допомогою методу рухомого фронту [1].

Алгоритм на основі генерації точок-вершин правильних шестикутників. Даний алгоритм безпосередньо для програмної реалізації є авторським. Його ресурсозатратність відносно велика [5], в середньому $O(n^2)$, що доведено на основі обчислювальних експериментів. Він нескладно реалізується в програмному середовищі C# [3, 4] з використанням класів та структур, проте вузлове покриття залежить від координат початкової точки, яка задається користувачем. Схематично алгоритм складається з ряду нижчевказаних кроків.

Користувачем вказується довільна точка, яка належить області Г. Дана точка приймається за перший вузол. Навколо цього вузла утворюються нові, які є точками правильного шестикутника з вказаною довжиною сторони R і з 6-ма точками на його межі. Описана дія проводиться рекурсивно. Тобто, навколо створених раніше вузлів утворюються нові, формуючи при цьому шестикутник навколо існуючого з довжиною сторони, рівній $2R$ і з 12-ма точками на його межі. Створюються нові вузли, відсікаючи ті, які не належать об-

ласті Г до тих пір, поки на $n-$ому кроці жоден із згенерованих вузлів не попаде в область Г. Вузли на межі області Г генеруються за допомогою методу рухомого фронту [1].

Алгоритм покриття вузлами методом рухомого фронту. Його перевагою є можливість згущення вузлів в потрібних місцях за допомогою спеціальної функції калібрування [1]. Даний метод відносно ресурсозатратний, алгоритм доволі складно реалізовується в середовищі програмування C#. На даний час не виведено єдиних критеріїв для знаходження функції калібрування [2]. В основі його лежить вузлація від межі області Г до її центру, при цьому регіон області, який в деякий момент часу вже вкритий вузлами (фронт), відмічається певним способом для унеможливлення подальшого внесення нових вузлів в даний регіон. Завершення роботи алгоритму відбувається при умові повного покриття області (фронт повністю вкриває область Г). Розглянемо покроковий алгоритм даного методу.

1. Нехай задана деяка замкнена многокутна область Г. Паралельно в оперативній пам'яті комп'ютера створюється растрове зображення для позначення фронту. *2.* На одній зі сторін області Г утворюються вузли шляхом перетину сторони з колами (з центрами на кінцях сторін та заданим радіусом R). При цьому на растровому зображенні даний фронт зафарбовується в чорний колір. *3.* З утворених вузлів утворюються нові аналогічно. Це продовжується допоки відстань між вузлами стає меншою за $2R$. Тоді утворюється новий вузол посередині між останніми. При цьому на зображеннях зафарбовується відповідний фронт. *4.* Таким чином утворюються вузли на всіх сторонах та зафарбовується на зображенні фронт. *5.* З двох сусідніх вузлів утворюємо новий за допомогою знаходження точки перетину кіл в цих вузлах з заданим радіусом R. При цьому потрібно перевірити на належність утвореної точки вже заповненому фронту з растрового зображення за допомогою функції взяття кольору. Якщо утворений вузол належить існуючому фронту, вузол відкидається. *6.* Аналогічно утворюємо вузли з вже знайдених, допоки вся область не буде вкрита вузлами (весь фронт на зображенні має заповнити многокутник).

Спрощений алгоритм рухомого фронту. Реалізація та розробка алгоритму є авторською, на основі методу рухомого фронту [1]. Даний алгоритм не передбачає можливості згущення вузлів і в центрі області генерує нерівномірне покриття. В порівнянні з алгоритмом рухомого фронту даний алгоритм швидше реалізовується в середовищі програмування C#, оскільки майже не містить математичних обчислень. Розглянемо алгоритм.

1. Нехай задана деяка замкнена многокутна область Г. Паралельно в оперативній пам'яті комп'ютера створюється растрове зображення для позначення фронту і його полотно повністю зафарбовується в чорний колір.

2. На утвореному растровому зображенні заповнюємо полігон фігури в білий колір за допомогою стандартної функції C# FillPolygon.

3. Після цього в циклі рядково з верху до низу і з низу до верху, прямуючи з обох боків до середини зображення, за допомогою стандартної функції GetPixel перевіряється колір пікселя.

4. Якщо колір пікселя чорний, даний кандидат у вузли до уваги не береться, якщо ж колір пікселя білий, даний вузол вноситься в список вузлів.

5. При перевірці всіх кандидатів у вузли виконується вивід згенерованих вузлів.

Покриття області вузлами за допомогою різних алгоритмів із вказанням часу роботи зображено, відповідно, на рис. 1-4. Характеристики персонального комп'ютера, на якому проводились обчислювальні експерименти: OS Microsoft Windows XP Professional Service Pack 3, CPU Pentium 4(R) 3.06GHz, RAM 2.00 Gb.

Як вже наголошувалось вище, задачі вузлації та тріангуляції взаємопов'язані. Розглянемо **задачу тріангуляції Делоне** за існуючими точками. Нехай маємо деяку скінченну множину точок $P_i = (x_i, y_i), i = \overline{1,n}$, які є вершинами замкненої однозв'язної многокутної області Г. Припустимо, що даний многокутник попередньо вкритий вузлами загальною кількістю m (дане вузлове покриття має включати в себе і всі точки $P_i, i = \overline{1,n}$). На основі відомого вузлового покриття можна згенерувати скінченноелементне покриття області, використовуючи **алгоритми тріангуляції Делоне**. В основу даних алгоритмів покладена так звана умова Делоне – будь-який вузол не повинен входити в область описаного кола навколо окремо взятого трикутника, крім вузлів самого трикутника.

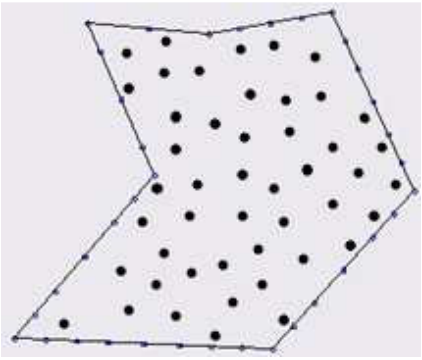

Рис. 1. Вузлація алгоритмом випадкового покриття, час роботи 0,08655439 с

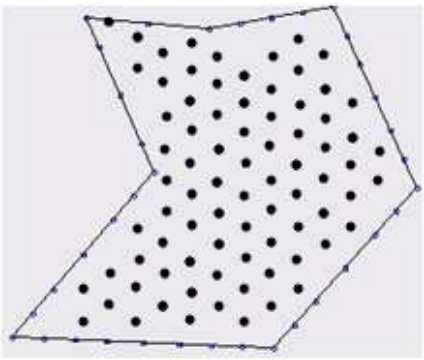

Рис. 2. Вузлація на основі генерації правильних шестикутників, час роботи 0,07367552 с

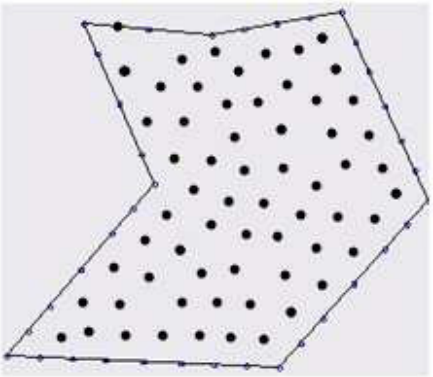

Рис. 3. Вузлація алгоритмом рухомого фронту, час роботи 0,04526258 с

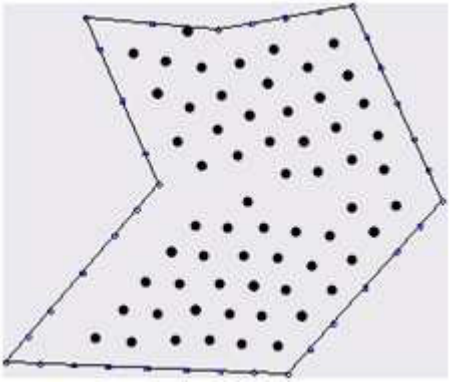

Рис. 4. Вузлація спрощеним алгоритмом рухомого фронту, час роботи 0,1154893 с

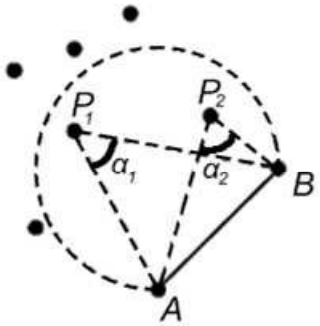

Рис. 5. Демострація покрокового алгоритму

Розглянемо кілька відомих алгоритмів тріангуляції Делоне [6].

Покроковий алгоритм, відомий також як алгоритм прямого перебору та метод активних ребер. Його трудомісткість складає $O(n^2)$. Він чудово реалізовується в середовищі програмування C#, проте є ресурсозатратним, оскільки включає в себе багато викликів пошуку точки. Розглянемо його.

1. Вибирається деяка базова лінія AB, починаючи з якої будуть будуватися всі наступні трикутники. Вона береться, як одна зі сторін багатокутника опуклої оболонки всіх вихідних точок тріангуляції.

2. Для базової лінії необхідно знайти сусіда Делоне – вузол, який разом з кінцями даної базової лінії в тріангуляції Делоне є вершинами одного трикутника. Процес пошуку можна представити як зростання «бульбашки» від базової лінії, поки не зустрінеться деякий вузол. «Бульбашка» – це коло, яке проходить через точки A і B (див. рис. 5), і центр якого знаходиться на серединному перпендикулярі до базової лінії.

3. Потрібно вибрати серед усіх точок P тріангуляції таку, що кут APB буде максимальним (наприклад, на рис. 5 буде обрана точка P_2).

4. Знайдений сусід Делоне з'єднується відрізками з кінцями базової лінії і утворює трикутник APB.

5. Нові ребра AP та BP побудованого трикутника позначаються як нові базові лінії, процес пошуку трикутників продовжується.

Інкрементний однопрохідний алгоритм. Він був представлений на науковій конференції «Pan Pacific Computer Conference» [7], яка проходила в січні 1989 року в Пекіні. Алгоритм чудово реалізовується в середовищі програмування C#, використовуючи принципи об'єктно-орієнтованого програмування. Використовує мало перевірок та пошуків елементів. В середньому він швидший в 5 разів за покроковий алгоритм. В обох методах використовувався авторський алгоритм відсіювання трикутників, які не належать області (у випадках неопуклих областей). Схематично розглянемо інкрементний однопрохідний алгоритм, який графічно проілюстрований на рис. 6-7 і складається з наступних етапів:

1. Ініціалізація списку трикутників. *2.* Визначення сторін «супертрикутника» – трикутника, в якому повністю лежать всі вузли. *3.* Включення вершин супертрикутника до кінця списку вершин. *4.* Включення супертрикутника до списку трикутників. *5.* Перевірка кожної нової точки із списку вузлів на те, чи задовольняють умову Делоне сусідні трикутники з даною точкою. *6.* Якщо точка лежить за межами описаних кіл сусідніх трикутників, то ребра цих трикутників утворюють з даною точкою нові трикутники, які додаються до списку трикутників. *7.* Якщо умова пункту *6* не виконується, сусідні трикутники видаляються і потім утворюються нові на їх місці, які задовольняють умову Делоне. Це повторюється із всіма точками. *8.* Видалення трикутників, які містять сторони супертрикутника і сам супертрикутник. *9.* Кінець.

Приклад тріангуляції Делоне області (вузлація області проведена алгоритмом рухомого фронту) наведено на рис. 8.

Розглянемо алгоритм видалення трикутників у випадку тріангуляції неопуклих областей.

Алгоритм виключення. Даний алгоритм є авторським. Він досить легкий в реалізації на об'єктно-орієнтованій мові програмування C#. Кожен трикутник в тріангуляції є класом, який має окреме поле даних «геометричний центр» і який обчислюється в ході створення трикутника. Дане поле – це точка в двовимірному просторі, координати (значення абсциси та ординати) якої обчислюються середнім арифметичним відповідних значень абсцис та ординат вершин трикутника. Після тріангулювання невипуклих областей утворюється область тріангуляції, яка є випуклим многокутником, оскільки тріангулювання вищенаведеними алгоритмами не передбачає врахування топологічних особливостей області для тріангуляції. Далі почергово перебира-

ються всі геометричні центри трикутників і перевіряється належність їх даній області алгоритмом трасування променів [9] (алгоритм перевірки належності точки многокутній області). Трикутники видаляються зі списку трикутників, якщо їхні геометричні центри не належать області. Таким чином відсіюються трикутники, які не належать тріангуляції даної неопуклої області.

Основним завданням статті був огляд та вдосконалення деяких для вузлового покриття двовимірних областей. Однак, відомо, якщо область покрита вузлами, то існує можливість побудови трикутної сітки з вершинами в даних вузлах. Тому в статті паралельно ставилось завдання реалізації алгоритмів скінченноелементної тріангуляції за заданою вузловою сіткою. Вузлові та відповідні їм трикутні сітки будуть корисними, наприклад, при порівнянні наближених чисельних розв'язків крайових задач математичної фізики сітковими (метод скінченних елементів) та безсітковими методами (наприклад, метод радіальних базисних функцій).

Отже, нами: 1) зроблено огляд деяких алгоритмів вузлового покриття двовимірних однозв'язних багатокутних областей; 2) проведено огляд деяких алгоритмів тріангуляції Делоне; 3) запропоновано алгоритм видалення трикутників у випадку тріангуляції неопуклих областей; 4) програмно реалізовано розглянуті алгоритми з використанням сучасної технології візуально-подійного програмування на мові C#; 5) проведено ряд чисельних експериментів.

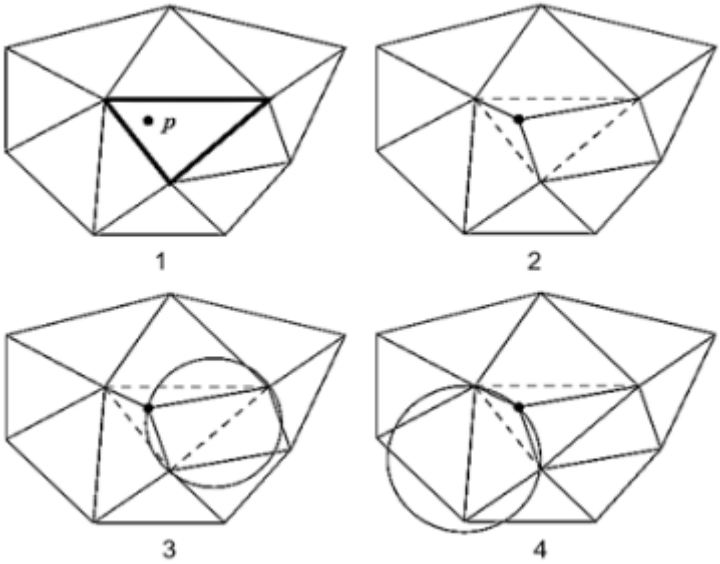

Рис. 6. Демострація інкрементного однопрохідного алгоритму

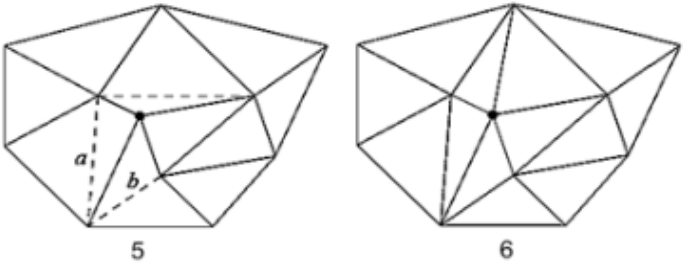

Рис. 7. Демострація інкрементного однопрохідного алгоритму

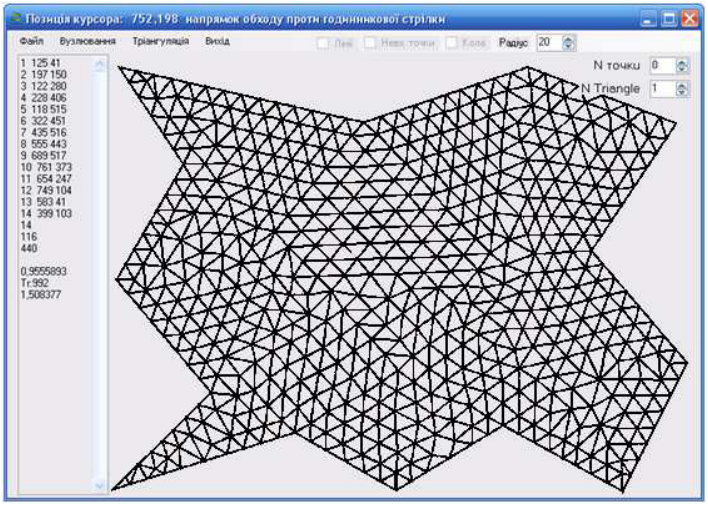

Рис. 8. Демонстрація тріангуляції Делоне

В **подальшому планується** дослідження існуючих та розробка нових алгоритмів вузлації та відповідного розбиття на скінченні елементи у випадку просторових областей.

1. Li T. Biting: Advancing front meets sphere packing / T. Li, A. Ungor // Int. J. Num. Meth. Eng. – 2000. – 49, №1-2. – P. 61-81. **2.** Li T. Biting ellipses to generate anisotropic meshes / T. Li, A. Ungor // Proc. 8th Meshing Roundtable. – P. 97-108. **3**. Дрейер М. C# для школьников. / М. Дрейер. – М.: БИНОМ, 2009. – 128 с. **4**. Шилдг Г. Полный справочник по C#. / Г.Шилдг.– М.: Вильямс, 2004. – 752 с. **5**. http://bestcode.org/. **6**. Bourke P. Efficient Triangulation Algorithm Suitable for Terrain Modellingor [Електронний ресурс]. – Режим доступу: http://www.codeguru.com/. **7**. [Електронний ресурс]. – Режим доступу: http://algolist.ru/. **8**. Liu G. R. Mesh Free Methods: Moving beyond the Finite Element Method / G. R. Liu. – Boca Raton: CRC Press, 2009. – 872 p. **9**. http://ru.wikipedia.org/wiki/Алгоритм_точка_в_многоугольнике/

Рецензент: д.т.н., професор Власюк А.П. (НУВГП)