

User name:
Volodymyr Matiiivskyi

Check date:
09.01.2023 14:10:33 EET

Report date:
09.01.2023 14:15:20 EET

Check ID:
1013386860

Check type:
Doc vs Internet + Library

User ID:
100010994

File name: **Цалапов А. Ё**

Page count: **51** Word count: **7606** Character count: **58597** File size: **1.65 MB** File ID: **1013158886**

2.68% Matches

Highest match: **1.5%** with Internet source (<https://www.bibliofond.ru/view.aspx?id=787505>)

1.89% Internet sources 3

Page 53

0.79% Library sources 1

Page 53

0.43% Quotes

Quotes 2

Page 54

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters 23

ВСТУП

Більшість чисельних методів дослідження напружено-деформованого стану деформованого тіла базуються на ідеї переходу від континуальної задачі до дискретної, коли досліджувана суцільна область замінюється деякою скінченною дискретною моделлю. У методі скінчених елементів (МСЕ) неперервна область замінюється деякою сукупністю скінчених елементів, що заповнюють весь об'єм тіла.

Однією з головних проблем, що виникають при застосуванні МСЕ - це побудова дискретної моделі досліджуваної механічної системи. Однією з головних частин будь-якого програмного комплексу чисельного аналізу є програма, яка автоматизує побудову геометричної моделі досліджуваного об'єкта з подальшою її дискретизацією на скінчені елементи.

Проблема оптимальної дискретизації досліджуваної області на скінчені елементи в загальному вигляді є досить складною (особливо для тривимірних областей). Це обумовлено тим, що на форму скінчених елементів (СЕ) накладаються два основних обмеження: вони не повинні мати надто малих (або відповідно занадто великих) кутів і обсяг СЕ не повинен перевищувати деяку задану величину. У першому випадку при розрахунках виникають значні обчислювальні похибки. У другому з'являється ризик втрати точності обчислень при значній зміні градієнта досліджуваної функції.

Тому автоматична генерація СЕ- сітки являє собою досить складну процедуру, яка є основою будь-якого скінчено-елементного пакету програм і являється актуальною.

Об'єкт дослідження – дискретизація геометричних областей на скінчені елементи.

Предмет дослідження – генерація розрахункових сіток для скінчено – елементного моделювання конструкцій.

Мета роботи – аналіз методів дискретизації геометричних областей на скінченні елементи заданої форми та розробка додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

Методи дослідження: методи обчислювальної математики і комп'ютерної графіки.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати основні поширені методи й алгоритми дискретизації плоских та просторових областей;
- провести моделювання та аналіз програмного забезпечення додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій;
- розробити і реалізувати додаток для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

Практичною цінністю роботи є розроблений додаток для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

В першому розділі проаналізовано основні поширені методи й алгоритми дискретизації плоских та просторових областей. дискретизації області на скінченні елементи.

В другому розділі виконано аналіз процесу розробки додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій. Проведено моделювання та описано архітектуру розробленого додатку.

У третьому розділі аргументується вибір середовища розробки додатку і мови програмування. Описано алгоритм генерації сіток. У додатку доступно створення 1- 2 - і 3-х мірних сіток, і їх оптимізація. Програма реалізується на мові програмування C++ в середовищі Visual C++ 2010. При створенні засобів відображення та візуалізації використовується графічний інтерфейс OpenGL.

РОЗДІЛ 1. ДИСКРЕТНІ (СКІНЧЕННО-ЕЛЕМЕНТНІ) МОДЕЛІ КОНТИНУАЛЬНИХ ОБЛАСТЕЙ

Більшість чисельних методів дослідження напружено-деформованого стану тіла, базуються на ідеї переходу від континуального завдання до дискретного, коли досліджувана суцільна область замінюється деякою кінцевою дискретною моделлю. У МСЕ безперервна область замінюється деякою сукупністю кінцевих елементів, що заповнюють увесь об'єм тіла, що не перетинається.

Одна з головних проблем, що виникають при застосуванні МСЕ, - це побудова дискретної моделі досліджуваної механічної системи. Тому, однією з головних частин будь-якого програмного комплексу чисельного аналізу, як уже згадувалося, є препроцесор - програма, що автоматизує побудову геометричної моделі досліджуваного об'єкту з подальшою її дискретизацією на кінцеві елементи. Від якості препроцесора багато в чому залежить і якість усього програмного комплексу в цілому.

Проблема оптимальної дискретизації досліджуваної області на кінцеві елементи в загальному вигляді є дуже складною (особливо для тривимірних областей). Це обумовлено тим, що на форму СЕ накладаються два основні обмеження : вони не повинні мати занадто малих (чи відповідно занадто великих) кутів і об'єм СЕ не повинен перевищувати деяку наперед задану величину. У першому випадку при розрахунках виникають значні обчислювальні погрішності. У другому з'являється ризик втрати точності обчислень при значній зміні градієнта досліджуваної функції (наприклад, в зоні передбачуваного концентратора напруги).

Тому автоматична генерація СЕ -сітки є дуже складною процедурою, що є основою будь-якого кінцево-елементного пакету програм. На практиці частіше використовуються СЕ у формі трикутника, прямокутника, тетраедра або паралелепіпеда, оскільки вони дозволяють з високою мірою точності апроксимувати область довільної форми.

1.1. Методи геометричного моделювання складних об'єктів

1.1.1. Опис топології області

Одним з найважливіших елементів чисельного розрахунку напружено-деформованого стану (НДС) тіла, що деформується, є побудова адекватної геометричної моделі досліджуваної області. Як правило, на практиці доводиться мати справу з об'єктами дуже складної конфігурації, що істотно ускладнює побудову таких моделей. В той же час від точності побудованої геометричної моделі багато в чому залежатиме якість отриманого чисельного результату.

Нині існують різні способи опису геометрії модельованої області. Одним з найчастіше використовуваних підходів являється використання спеціалізованих CAD -систем, що дозволяють побудувати необхідну топологічну модель, як деяку сукупність базових геометричних примітивів. Такий підхід застосовується, наприклад, в системах ANSYS, COSMOS і COSAR [12]. У препроцесорах цих систем є бібліотеки таких графічних примітивів, як точка, лінія, сплайн, ламана, коло, сфера, конус, куб та ін., над якими визначені ейлереві операції їх об'єднання, перетини і віднімання, що дозволяють задати практично довільну область. Альтернативним є підхід, що полягає в параметричному описі геометрії модельованої області за допомогою деякої мови опису топології області. Наприклад, система геометричного моделювання NETGEN [12] використовує для опису топології області мову CSG (ConstructiveSolidGeometry), що дозволяє описувати невеликі і середні плоскі і тривимірні області. У CSG в текстовому форматі ASCII можна описувати довільну просторову область, як логічну комбінацію наступних базових геометричних примітивів:

- площина;
- циліндр;
- сфера;
- еліптичний циліндр;
- еліпсоїд;

- конус;
- паралелепіпед;
- многогранник.

Геометрія об'єкту визначається як деяка сукупність ейлеревих операцій (об'єднання, перетин і доповнення) над вищеописаними примітивами.

Іншим поширеним способом опису топології тривимірних об'єктів є так званий формат стерео літографії (STL - формат). Цей формат застосовується в автоматизованих системах проектування для опису тривимірних моделей і є для них найбільш часто-використовуваним стандартним форматом.

Інформація про об'єкт в STL – файлі включає список трикутних граней, які описують поверхню його твердотілої моделі із заданою точністю, і може бути представлена у вигляді текстового (ASCII) або бінарного файлу. Текстове представлення STL - файлу повинне починатися ключовим словом SOLID і закінчуватися ENDSOLID. Між цими програмними дужками наводиться опис трикутників. Опис кожного трикутника включає завдання одиничного вектору нормалі, спрямованого від його поверхні, після чого слідує список тривимірних координат усіх вершин. Усі координати представлені в ортогональній декартовій системі координат і записані у вигляді дійсних чисел.

На рис. 1.1 наведений приклад опису одного трикутника в STL -форматі:



Рис.1.1. Опис трикутника в STL -форматі.

При правильному описі тріангульованої поверхні усі сусідні трикутники повинні мати по дві загальні вершини.

1.1.2. Дискретизація плоских областей

Тріангуляцією плоскої області називається її розбиття на деяку сукупність трикутників, що не перетинаються. Усі поширені алгоритми автоматичної дискретизації областей на кінцеві елементи оперують поняттям тріангуляції Делоне [15]. Тріангуляцією Делоне називається безліч трикутників, що не перетинаються, для яких виконується умова: в коло, описане біля довільного трикутника, не потрапляє жодна вершина, що належить будь-якому іншому трикутнику (рис. 1.2).



Рис. 1.2. Приклади тріангуляції (а) і тріангуляції Делоне (б).

Також базовим поняттям в тріангуляції плоских областей є діаграма Воронова. Діаграмою Воронова для деякої безлічі точок на площині називається сукупність полігональних (многокутних) фігур, утворених лініями, які перпендикулярні відрізка, що сполучають задані точки (рис. 1.3).

Нині розроблена велика кількість алгоритмів автоматичної генерації тріангуляції. Огляд найбільш поширених алгоритмів приведений в роботі. Серед них можна виділити такі:

- алгоритм Ватсона;
- алгоритм Лавсона;
- комбінований алгоритм Ватсона і Лавсона;
- алгоритм послідовного розбиття;
- алгоритм ділення і включення;
- покроковий алгоритм;
- модифікований ієрархічний алгоритм;
- алгоритм Рапперта.



Рис. 1.3. Діаграма Воронова.

Більшість з перерахованих алгоритмів базуються на ідеї побудови триангуляції Делоне для заданої на площині сукупності точок. Одним з найбільш ефективних і легких в реалізації являється комбінований алгоритм Ватсона і Лавсона. Його суть полягає в наступному. Нехай на площині задана деяка сукупність точок. Триангуляційна процедура послідовно вставляє кожен наступну точку, починаючи з першої, у вже існуючу триангуляцію. Спочатку триангуляція Делоне представлена одним єдиним так званим супертрикутником, усередині якого на початковому етапі розташовується уся задана сукупність точок. Для цього, наприклад, координати точок нормуються так, щоб вони лежали на інтервалі від 0 до 1, а координати вершин супертрикутника приймаються рівними $(-100, -100)$, $(100, -100)$ і $(0, 100)$.

При розгляді чергової точки P , в першу чергу, знаходиться трикутник, що містить P , а потім будуються три нові трикутники шляхом з'єднання точки P з вершинами трикутника, що обгороджує її. Після цього початковий трикутник, що обгороджує точку P видаляється, і загальне число побудованих трикутників збільшується на два.

Після обробки точки P отримане для неї розбиття перетворюється в триангуляцію Делоне за допомогою обмінного алгоритму Лавсона. У цій процедурі всі трикутники, суміжні з протилежними до точки P ребрами, поміщаються в стек (максимальний трикутник поміщається в стек першим). Кожен поміщений в стек трикутник, піддається перевірці, в ході якої визначається, чи лежить P за межами кола, описаного біля тестованого трикутника. Якщо це випадок, коли P є вершиною трикутника, і суміжний

трикутник утворює з даним опуклий чотирикутник, в якому діагональ проведена неправильно, то діагональ проводиться по-іншому. В результаті обмінної процедури Лавсона і робиться перетворення отриманої триангуляції в триангуляцію Делоне.

Обмінна процедура міняє два старі трикутники на два нових. Після одного обміну усі протилежні до точки P трикутники додаються в стек (це максимум два трикутники). Наступний трикутник виштовхується із стека, і увесь процес повторюється, поки стек не стане порожнім. Після цієї фази для точки P виходить нова триангуляція Делоне. Суть обмінної процедури Лавсона приведена на рис. 1.4. Тут слід зауважити, що якщо точка P лежить поза межами кола, то необхідно перейти до наступного трикутника в стеку.

Лавсоном було показане, що цей ітераційний алгоритм повинен побудувати триангуляцію Делоне і завершитися після останнього обміну. Практика показує, що для побудови Делоне-триангуляції не потрібно велику кількість обмінів, тому цей процес є ефективним.

Після того, як до триангуляції будуть додані усі точки, підсумкова триангуляція Делоне виходить шляхом видалення усіх трикутників, що мають в якості вершин вершини супертрикутника. Будь-яка вершина трикутника, що видаляється, не співпадаючи з вершиною супертрикутника, повинна лежати на межі триангуляції.

Оскільки вставка кожної нової точки в триангуляцію створює два нові трикутники, загальне число отриманих трикутників в триангуляції має дорівнювати $2N+1$, де N - число вершин, що беруть участь в триангуляції.

Тести показують, що для N довільно розташованих на площині точок розрахунковий час алгоритму складає $O(N^{5/4})$. Крім того, для роботи алгоритму потрібно близько $14N$ елементів пам'яті, використовуваних для зберігання координат точок, номерів вузлів трикутників і іншої допоміжної інформації [13].

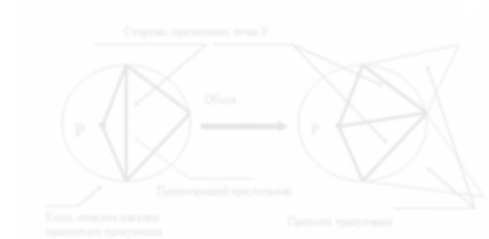


Рис. 1.4. Обмінний алгоритм Лавсона.

Загальна блок-схема комбінованого алгоритму Ватсона-Лавсона приведена на рис. 1.5. Для його застосування в першу чергу необхідно задати межу області, що підлягає триангуляції, а потім опорні точки, на яких і буде побудована триангуляція. Після чого, з метою підвищення ефективності роботи алгоритму, координати опорних вузлів нормуються і сортуються. Після триангуляції початкові координати вузлів відновлюються.



Рис. 1.5. Загальна блок-схема комбінованого алгоритму Ватсона-Лавсона.

Головною труднощію, що виникає при використанні цього алгоритму, є пошук і видалення зайвих трикутників при триангуляції багатозв'язкових або неопуклих областей (рис. 1.6).

Ця проблема зводиться до рішення задачі про належність точки (наприклад, геометричного центру трикутника) заданому багатокутнику (межі області).



Рис. 1.6. Проблема зайвого трикутника.

Відома велика кількість методів і алгоритмів рішення цієї задачі. Найбільш ефективними серед них є наступні:

- підрахунок кількості перетинів межі області променем, проведеним з тестованої точки (рис. 1.7);



Рис. 1.7. Визначення належності точки замкнутому контуру шляхом підрахунку кількості перетинів променя, проведеного з тестованої точки, і межі області.

- визначення величини кута, утвореного відрізками, що сполучають тестовану точку і сусідні вершини контура (рис. 1.8).



Рис. 1.8. Визначення належності точки замкнутому контуру шляхом підрахунку суми кутів.

У першому випадку, якщо точка знаходиться усередині області, то кількість перетинів променя має бути непарною. А в другому - якщо точка

знаходиться усередині контура, сумарний кут повинен дорівнювати 360° якщо на межі - 180° Інакше точка знаходиться за межами області.

Проблема побудови безлічі опорних точок для триангуляції Делоне в загальному вигляді є досить складною, оскільки необхідно враховувати як обмеження, що накладаються на форму трикутників, так і можливу наявність в геометрії триангульованої фігури так званих сингулярностей : тріщин, розрізів, отворів, гострих кутів і тому подібне, що вимагає значного згущування сітки в їх області.

Одним з найбільш ефективних алгоритмів побудови триангуляції Делоне, що дозволяють здолати описані вище проблеми, являється алгоритм Рапперта. Фактично цей алгоритм оптимізаційний. Він дозволяє поліпшити якість вже наявного первинного розбиття.

Ідея алгоритму Рапперта полягає в наступних двох кроках:

- а) отримання первинної («грубої») триангуляції плоскої області шляхом завдання деякої сукупності точок на її межі;
- б) оптимізація цієї триангуляції шляхом введення в сітку нових вузлів з подальшим перетворенням розбиття в триангуляцію Делоне.

Поліпшення якості кінцево-елементної мережі досягається за рахунок розбиття трикутників, що мають неправильну форму (гострі кути або велика площа) шляхом введення нових точок. При цьому величини кутів і площа елементів є параметрами алгоритму, що дозволяють управляти процесом розбиття.

При описі алгоритму Рапперт ввів терміни[16], що фактично стали в теорії триангуляції загальноприйнятими :

- елемент (element) - трикутник;
 - сегмент (segment) - відрізок, що сполучає сусідні точки, що лежать на межі області;
 - вузол (node) - точка, в якій сходяться ребра дотичних елементів.
- Вузлам відповідають точки на діаграмі Воронова (рис. 1.3);

- ребро (edge) - відрізок, по якому граничать сусідні елементи;
- включена точка (encroachedpoint) - довільна точка поточного розбиття, що знаходиться усередині кола, радіусом якого є довільний сегмент;
- «неправильний» трикутник (badtriangle) — елемент з характеристиками (кути, площа) що не задовольняють заданим обмеженням на триангуляцію.

Алгоритм Рапперта складається з двох базових процедур:

- 1) розбиття «неправильного» трикутника шляхом введення нового вузла;
- 2) розбиття сегментів шляхом введення нового вузла.

Розбиття «неправильного» трикутника відбувається таким чином:

- а) обчислюються координати кола, описаного біля елемента, який підлягає розбиттю;
- б) у центр кола додається новий вузол;
- в) початковий елемент віддаляється і замінюється знову утвореними (рис. 1.9).



Рис. 1.9. Розбиття «неправильного» трикутника.

Розбиття сегменту відбувається у тому випадку, якщо в коло, діаметром якого він є, потрапляє вузол (рис. 1.10), що не належить йому. Тоді такий «неправильний» сегмент ділиться таким чином:

- сегмент ділиться навпіл;
- у середину сегменту додається новий вузол;
- видаляється трикутник, ребром якого був початковий граничний сегмент;
- будуються нові трикутники.

Таким чином, побудова триангуляції Делоне з використанням алгоритму Рапперта полягає в послідовному переборі усіх елементів і їх оптимізації із застосуванням двох базових процедур.



Рис. 1.10. Розбиття «неправильного» сегменту.

Оригінальний алгоритм Рапперта містить тільки один параметр - критерій якості згенерованої сітки (мінімальний кут сітки). Він визначався як мінімальний по усіх елементах кут між ребрами. Проте, при використанні МКЕ потрібно введення ще одного критерію якості сітки - максимальної площі елемента. Тому усі сучасні реалізації і модифікації алгоритму Рапперта використовують ці два критерії. Раппертом було показано, що критерій мінімального кута розбиття автоматично забезпечує згущування сітки поблизу сингулярностей, які, як правило, є концентраторами напруги.

Таким чином, загальну блок-схему застосування алгоритму Рапперта для автоматичної оптимізації триангуляції можна зображувати таким чином (рис. 1.11).

Раппертом було також показано, що алгоритм буде стійкий і правильно працювати для мінімальних кутів $\alpha \approx 20^\circ$.

У оригінальному алгоритмі Рапперта явно не є присутньою процедура оптимізації якості сітки, що базується на повороті діагоналі. Суть цієї операції, як і в алгоритмі Ватсона-Лавсона, полягає в тому, що два суміжні трикутники, що мають загальну грань, утворюють чотирикутник, діагональ в якому можна провести різними способами, як це показано на рис. 1.12.

Поворот діагоналі можна використовувати для локальної оптимізації отриманого звичайно-елементного розбиття. Проте його використання вимагає наявності критерію, згідно з яким необхідно проводити цю процедуру. Найчастіше в якості такого критерію розглядаються площі

елементів до повороту діагоналі і після, а також критерій мінімального кута в елементах.

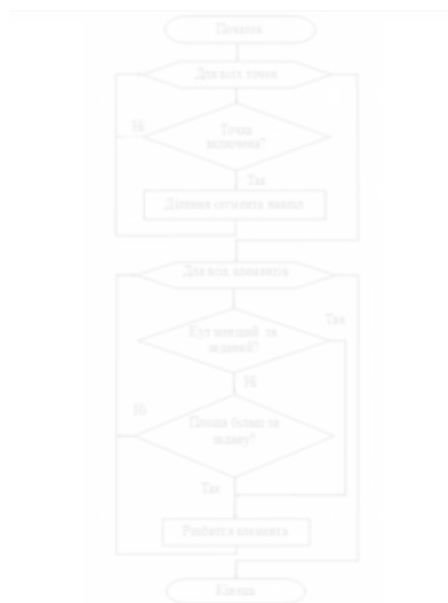


Рис. 1.11. Блок-схема алгоритму Рапперта.

Поворот діагоналі зручно проводити відразу ж після виконання однієї з двох базових процедур алгоритму Рапперта. В цьому випадку точно відома область, для якої були проведені зміни в сітці, що дозволяє швидко знайти потрібні елементи і при необхідності виконати поворот діагоналі.



Рис. 1.12. Процедура повороту діагоналі.

При застосуванні алгоритму Рапперта можливе виникнення наступних проблем. При виконанні розбиття рівнобедрених трикутників можуть виникати нові елементи з нульовою площею. Це пояснюється тим, що при попаданні центру описаного навколо трикутника кола на одну з його сторін площа одного з знову утворених трьох трикутників дорівнюватиме нулю. З

рис. 1.13 видно, що після розбиття **трикутника ABC** утворюються три нових: AOB , BCO і вироджений AOC .



Рис. 1.13. Утворення виродженого трикутника.

Одним із способів подолання цієї проблеми є додавання нової точки при розбитті трикутника не строго в центр описаного кола, а в деяку її околицю. Іншим поширеним способом є виключення виродженого трикутника з мережі.

Другою проблемою є розбиття трикутника, центр описаного біля якого кола лежить за його межами. В цьому випадку виникає проблема зациклення, оскільки точка, що додається, не змінює форму трикутника. Одним з можливих варіантів вирішення цієї проблеми є перевірка на попадання точки, що вставляється, в трикутник. Якщо точка лежить за його межами, то трикутник розбивається, наприклад, шляхом ділення навпіл його найбільшої сторони (рис. 1.14).



Рис. 1.14. Розбиття витягнутого трикутника.

Наступною відомою проблемою, являється дискретизація областей, в геометрії яких є присутніми гострі кути. В цьому випадку можливе зациклення процедури ділення граничних сегментів, як показано на рис 1.15.



Рис. 1.15. Зациклення процедури ділення сегменту при гострому вуглі.

Раппертом був запропонований спосіб вирішення цієї проблеми, що полягає в тому, що «неправильний» сегмент при гострому вуглі ділиться не посередині, а в найближчій до середини точці перетину цього сегменту і концентричних кіл з центрами у вершині кута (рис. 1.16). Причому радіус кожного кола в два рази більший за попередній, а радіус першого довільний, але береться досить малим відносно довжини «неправильного» сегменту.



Рис. 1.16. Ділення сегменту способом концентричних кіл.

Нині алгоритм Рапперта є однією з найбільш ефективних і часто використовуваних для Делоне-триангуляції плоских областей. Хоча, як було відмічено вище, формально він є усього лише алгоритмом оптимізації вже існуючого розбиття області. Тому для його застосування необхідно отримати первинну дискретизацію області на кінцеві елементи. Зробити це можна, наприклад, таким чином. На межі області з наперед заданим кроком задаються вузли, які будуть використані для побудови первинного «грубого» розбиття. Потім, використовуючи як опорні отримані вузли і застосовуючи послідовно модифікований алгоритм Ватсона-Лавсона і алгоритм Рапперта, можна безпосередньо отримати необхідну триангуляцію (рис. 1.17).



Рис. 1.17. Схема отримання початкового розбиття з подальшою триангуляцією.

1.1.3. Дискретизація тривимірних областей

Як і в плоскому випадку, дискретизація тривимірної області на кінцеві елементи розпочинається з початкового «грубого» розбиття. Поверхня

тривимірної області представляється як сукупність многокутних областей - полігонів. Іншими словами, спочатку відбувається дискретизація поверхні тривимірної області.

Потім, практично так само, як і в двовірному випадку, усі вершини триангульованої поверхні тривимірної області вставляються всередину супертетраедра.

Дж. Шевчук запропонував модифікацію алгоритму Рапперта для тривимірного випадку. Ним, по аналогії з Раппертом, була введена наступна термінологія:

- елемент - тетраедр;

- граничний сегмент (boundarysegment) - це ребро, що належить поверхні початкового об'єкту, який підлягає дискретизації, і задане у вхідному описі геометрії області; якщо граничний сегмент розділений на частини послідовною вставкою точок, кожна його частина називається граничним підсегментом;

- гранична грань (boundaryfacet) - це плоский багатокутник (полігон), що лежить на поверхні початкового об'єкту і обмежений граничними сегментами; Якщо гранична грань розділена триангуляцією або додаванням всередину нових точок, то її внутрішні частини називаються граничними підгранями;

- екваторіальна сфера (equatorialsphere) трикутної граничної грані або підграні - це єдина сфера, екватором якої є коло, описане біля заданого трикутника.

Алгоритм Дж. Шевчука у загальних рисах містить наступні кроки:

- 1) якщо тетраедр має «неправильну» форму, тобто його найкоротша грань занадто мала в порівнянні з радіусом описаної біля нього сфери, то вставляється новий вузол в центр цієї сфери, і елемент ділиться по аналогії з «неправильним» трикутником в алгоритмі Рапперта;

2) якщо вершина, що вставляється, потрапляє на граничну підгрань, то вона розділяється; підграні діляться шляхом вставки вузла в центр описаних біля них екваторіальних сфер;

3) якщо вузол, що вставляється, потрапляє на граничний підсегмент, то він також розділяється.

Шевчук визначив наступний пріоритет виконання цих операцій :

а) ділення підсегменту;

б) ділення підграні;

в) ділення елементів, що мають «погану» форму.

По аналогії з двовимірним випадком оптимізації отриманої сітки, коли виконується процедура повороту діагоналі, в тривимірному випадку також виконується обмінна процедура реконфігурації розбиття.

У тривимірному випадку ця процедура є складнішою. У канонічному випадку відбувається заміна двох суміжних тетраедрів на три або двох тетраедрів на два. Можлива і зворотна процедура (рис. 1.18).

Обмінна процедура для ребер є складнішою. У ~~ній~~**тетраедрів**, інцидентних єдиному ребру, замінюються новим набором з $2N - 4$ тетраедра. На рис. 1.19 наведений приклад **ребраAB**, перпендикулярного площині сторінки. П'ять тетраедрів, спочатку інцидентних йому (01**AB**, 12**AB**, 23**AB**, 34**AB** і 40**AB**), в результаті обміну були замінені шістьма новими тетраедрами, два для кожного з трикутників, що утворюють тріангуляцію полігону 01234: 012A, 024A, 234A, 021B, 042B і 324B. Після завершення цих процедур (як і в плоскому випадку), з отриманого кінцево-елементного розбиття видаляються усі елементи, що мають в якості вершин вершини супертетраедра [13].



Рис. 1.18. Приклади обмінної процедури в тривимірному випадку.



Рис. 1.19. Приклад обмінної процедури для ребер.

1.2. Об'єктна модель геометричної області

1.2.1. Модель дискретного представлення області

Після відпрацювання процедури дискретизації плоскої або тривимірної області отримане дискретне представлення геометричної області описується об'єктною моделлю, приведеною на рис. 1.20. Вона містить інформацію про загальну кількість вузлів у розбитті області на СЕ, кількості граничних вузлів, ширині стрічки, використовуваної надалі для стрічкових методів рішення СЛАР, а також інформацію про координати вузлів і зв'язків в СЕ [14].



Рис. 1.20. Об'єктна модель дискретного представлення геометричної області.

Крім того, міститься додаткова інформація про графу зв'язків між вузлами області, також використовувана надалі для отримання компактного представлення матриці жорсткості.

Для генерації скінченно-елементного розбиття використовується алгоритм, що є комбінацією алгоритмів Ватсона-Лавсона і Рапперта-Шевчука. Основні етапи роботи цього алгоритму побудови кінцево-елементної моделі області наступні:

а) генерація кінцево-елементного розбиття, що включає :

- побудова точкової (каркасної) моделі області;
- початкову («грубу») дискретизацію області на СЕ;
- додавання нових вузлів в розбиття;

б) оптимізація отриманого розбиття, що складається з :

- оптимізації мінімальних кутів елементів (обмінна процедура);
- контролю максимальної площі або об'єму.

Процес дискретизації геометричної області розпочинається з побудови її точкової (каркасної) моделі. У цій моделі об'єкт представляється як сукупність вузлів, що лежать на його межі або поверхні і зв'язків між ними. На рис. 1.21 показана схема переходу від об'єкту до його каркасної моделі, а потім і до дискретної.



Рис. 1.21. Етапи дискретизації області : а) об'єкт; б) каркасна модель; в) дискретна модель.

У плоскому випадку побудова каркасної моделі розпочинається з процесу «інтерполяції» - заміни криволінійних граничних сегментів ламаними прямими (рис. 1.22). При цьому обчислюється кількість необхідних вузлів так, щоб забезпечити їх мінімальну кількість при заданому параметрі розміру елементу.



Рис. 1.22. Дискретизація криволінійного граничного сегменту.

При заміні криволінійного сегменту ламаною прямою шляхом вставки нових вузлів важливо контролювати довжину сегментів ламаної, щоб не втратити точність при моделюванні граничних особливостей топології початкової області. При цьому чим більше кривизна криволінійної граничної ділянки, тим більше вимагається для його інтерполяції сегментів ламаної.

Для реалізації процедури «інтерполяції» криволінійних граничних сегментів потрібна наявність алгоритму визначення координат чергової точки, що вставляється на межу. Одним з можливих варіантів реалізації такого алгоритму є обчислення кута між дотичною до граничного сегменту і прямої, що містить передбачуваний сегмент ламаної, що починається у вже наявній точці і закінчується у вузлі (рис. 1.23), що знову вставляється. При цьому величина кута (обчислюється на початку роботи препроцесора і залежить від геометрії початкової області і параметра, що визначає максимальну величину елементів. Обчислити цей кут можна, знаючи величину розміру місцевої особливості $lfs(localfeaturesize)$ в початковій точці $p(x_0, y_0)$, визначеною Раппертом як радіус мінімального кола з центром в точці p , яка торкається двох сегментів межі області, що не перетинаються. Для кожного типу криволінійних сегментів межі плоскої області обчислення lfs не представляє складності [10].



Рис. 1.23. Обчислення координат вузла, що вставляється на граничний сегмент.

Недоліком цього алгоритму є те, що при побудові точкових моделей областей з граничними сегментами, що мають малий радіус кривизни, можливі значні спотворення його топології, що виникають за рахунок того,

що за умовчанням розмір елементів приймається максимально можливим (рис 1.24).



Рис. 1.24. Спотворення геометрії області при великих значеннях кута α .

Цю проблему можна розв'язати тільки за рахунок згущування сітки, тобто зменшення значення кута α .

Таким чином, блок-схему алгоритму побудови каркасної моделі плоскої геометричної області можна зображувати таким чином (рис. 1.25).



Рис. 1.25. Блок-схема алгоритму початкової дискретизації межі області.

1.2.2. Дискретизація області на скінченні елементи

Як вже відзначалося, в якості вхідної інформації алгоритму Рапперта потрібно початкове «грубе» розбиття області на елементи. Для попереднього розбиття плоскої або об'ємної геометричній області використовується вищеописаний модифікований алгоритм Ватсона-Лавсона.

Цьому алгоритму в якості вхідної інформації потрібний набір вузлових точок, на базі яких і буде побудована початкова тріангуляція або тетраедризація. В якості такого набору точок безпосередньо використовуються вузли, що утворюють каркасну модель. Для оптимізації початкового розбиття можна використовувати модифікований алгоритм Рапперта-Шевчука.

Визначення вузлів, що належать межі плоскої або поверхні об'ємної області, відбувається в процесі роботи алгоритму оптимізації початкового розбиття і додавання нових вузлів в дискретизацію. Проте на практиці часто виникає необхідність визначення граничних вузлів і сегментів для вже існуючого розбиття (наприклад, при використанні геометрії, побудованої іншим препроцесором). В цьому випадку виділити поверхневі грані (у тривимірному випадку) можна виходячи з того міркування, що внутрішня грань завжди належить двом СЕ (рис. 1.26). У плоскому випадку побудувати межу можна виходячи з аналогічних міркувань.



Рис. 1.26. Схема виділення поверхневих граней СЕ-об'єкту.

Багато вживаних спільно з МСЕ методи рішення СЛАР вимагають наявності додаткової інформації про зв'язки між вузлами кінцево-елементного розбиття. Така інформація дозволяє оптимізувати потрібний для зберігання коефіцієнтів матриці жорсткості об'єм оперативної пам'яті ЕОМ.

Для побудови списку зв'язків кожного вузла сітки з іншими на етапі дискретизації області необхідно побудувати модель розбиття у вигляді графа, а потім її досліджувати.

1.3. Висновки до розділу

В цьому розділі проаналізовано основні підходи, що застосовуються в сучасних САПР для твердотілого моделювання геометричних об'єктів, а

також основні поширені методи й алгоритми дискретизації плоских та просторових областей. Найпоширеніші алгоритми дискретизації можна розбити на дві частини: побудова первинної дискретизації (найбільш складний етап), та її оптимізація. У додатку автоматичної генерації розрахункових сіток для скінченно- елементного моделювання конструкцій при первинній дискретизації області на скінченні елементи прийнято застосувати модифікований алгоритм Ватсона-Лавсона. Оптимізацію отриманої скінченно-елементної сітки робити шляхом застосування алгоритму Рапперта в плоскому випадку та Шевчука – в тривимірному.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДОДАТКА ДЛЯ АВТОМАТИЗАЦІЇ ПОБУДОВИ ДИСКРЕТНОЇ (СКІНЧЕННО-ЕЛЕМЕНТНОЇ) МОДЕЛІ КОНСТРУКЦІЙ

Підходом до проектування різних систем, шляхом подання у вигляді діаграм їхніх статичних і динамічних моделей на всіх процесах життєвого циклу, являється мова візуального моделювання- UML (Unified Modeling Language).

В основу методу покладено парадигму об'єктного підходу, при якій концептуальне моделювання проблеми полягає у побудові [17]:

- онтології домену, яка визначає склад та ієрархію класів об'єктів домену, їх атрибутів і взаємозв'язків, а також операцій, які можуть виконувати об'єкти класів;

- моделі поведінки, яка задає можливі стани об'єктів, інцидентів, що ініціюють переходи з одного стану до іншого, а також повідомлення, якими обмінюються об'єкти;

- моделі процесів, що визначає дії, які виконуються при проектуванні об'єктів як компонентів.

Проектування в UML починається з побудови сукупності діаграм, які візуалізують основні елементи структури системи.

Мова моделювання UML підтримує статичні і динамічні моделі, зокрема модель послідовностей – одну з найкорисніших і наочних моделей, в кожному вузлі якої є взаємодіючі об'єкти. Всі моделі зображаються діаграмами, коротка характеристика яких дається нижче.

2.1. Логічне представлення статичної моделі структури програмної розробки

Повний проект програмної системи являє собою сукупність моделей логічного і фізичного уявлень, які повинні бути узгоджені між собою. У мові UML для статичного представлення моделей систем використовуються діаграми класів. Вони визначають склад класів об'єктів і їх зв'язків.

25

На діаграмі класів (Рис.2.1.) прямокутники, поділені на три частини – це класи, лінії – зв'язки між ними. Ім'я класу записано в верхній частині прямокутника. В другій і третій частині – відповідно список атрибутів і операції, що мають специфікатори доступу.

Специфікація класів сіткового генератора.

cPoint- клас вершин елементів в n -мірному просторі, який містить координати точки, її індивідуальний номер в списку вершин усіх точок.

cElement- клас елементів в просторі R_n . Цей клас зберігає в собі список номерів вершин, їх кількість, індивідуальний номер елементу, координати його центру мас, об'єм, номери сусідніх елементів, що мають з ним загальну грань. Для моделювання завдань, що мають об'єкти з різними фізичними властивостями використовується змінна, що містить номер підобласті, якій належить елемент. Крім того клас елементу містить методи для динамічного обчислення граней.

cFacet- клас граней елементу, який містить список номерів вершин, що утворюють грань; їх кількість, площу, нормаль, орієнтовану з елементу хазяїна грані, номер елементу сусіда хазяїна грані, і номер елементу хазяїна грані. Грань обчислюється динамічно у міру використання для економії пам'яті.

cMesh- клас для зберігання і доступу до сітки в тілі програми. Цей клас містить список точок і елементів сітки. Іноді такі класи містять список граней. Проте, як показує практика, зберігання списків граней призводить до значної перевитрати пам'яті. Цей клас містить у тому числі методи для: читання і запису сіток в різних сіткових форматах, методи для реалізації згущення і розрідження сіток, які використовуються для адаптивних сіткових методів, методи по перебудові сіток.

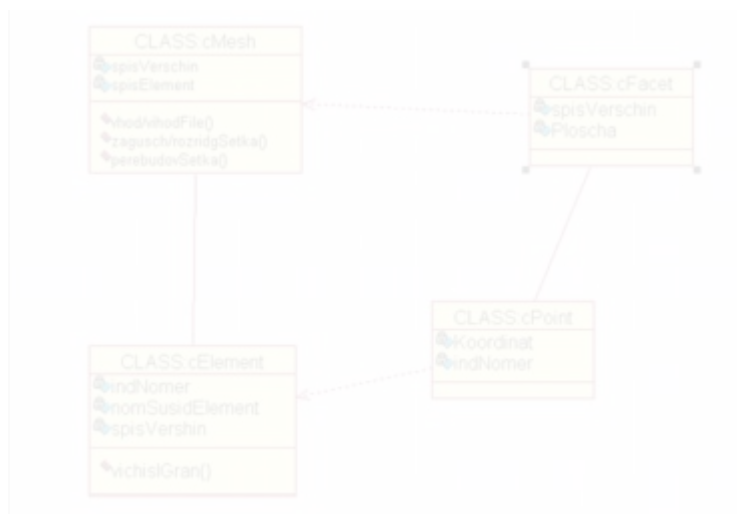


Рис.2.1. Діаграма класів сіткового генератора

2.2. Логічне представлення моделі поведінки програмної розробки

Поведінка програмної розробки визначається множиною об'єктів, що обмінюються повідомленнями.

На UML діаграмі варіантів використання (Use Case Diagram) (Рис.2.2.) показано взаємодію між варіантами використання і діючими особами. Вона відображає вимоги до системи з точки зору користувача. В нашому випадку варіанти використання - це функції, виконувані програмним комплексом, а дійові особи - це користувачі створюваного програмного забезпечення. Такі діаграми показують, які діючі особи ініціюють варіанти використання. З них також видно, коли дійова особа отримує інформацію від варіанту використання.

Специфікація Use Case Diagram «Дискретизація фігури» (Рис.2.2.)

Коротке описання:

Use Case Diagram дозволяє користувачу дискретизувати фігуру на 1D, 2D, 3D - скінчені елементи.

Користувачі системи:

1. Оператор- задає команди для побудови моделі.
2. Програмний комплекс-виконує команди оператора.

Варіанти використання:

1. Запуск програми.
2. Встановити масштаб.
3. Вибір моделі.
4. Задати вид моделі.
5. Задати тип дискретизації.
6. Розбиття фігури на скінчені елементи.
7. 1D- розбиття.
8. 2D- розбиття.
9. 3D- розбиття.
10. Дискретизована фігура.
11. Оптимізація сітки скінчених елементів.
12. Збереження сітки скінчених елементів.

Користувач системи «Оператор»

Коротке описання :

Даний користувач системи описує дії оператора для побудови моделі.

Основні події:

1. Оператор запускає програму.
2. Програма завантажується.
3. Оператор вибирає фігуру.
4. Програмний комплекс завантажує обрану фігуру.
5. Оператор задає вид моделі.
6. Оператор задає масштаб
7. Оператор задає тип дискретизації.
8. Програмний комплекс автоматизує послідовність дій оператора.
9. Програмний комплекс виконує поставлені задачі оператора.

Користувач системи «Програмний комплекс»

Коротке описання:

Даний користувач системи описує дії програмного комплексу щодо побудови моделі.

Основні події:

1. Отримав запит оператора, завантажується для роботи.
2. Оператор задає тип дискретизації.
3. Програмний комплекс розбиває фігури на скінчені елементи.
4. Оператор прагне покращити якість сітки.
5. Програма оптимізує сітку скінчених елементів.
6. Оператор зберігає побудовану модель.
7. Програмний комплекс зберігає сітку скінчених елементів.

Варіант використання «Запуск програми»

Коротке описання:

Даний варіант використання описує запуск програми для дискретизації.

Основні події:

1. Оператор відкриває програму .
2. Програмний комплекс завантажується.

Варіант використання «Вибір фігури»

Коротке описання:

Даний варіант використання описує вибір оператором фігури.

1. Оператор вибирає потрібну фігуру.
2. Програма відображає вибрану фігуру.

Варіант використання «Встановити масштаб»

Коротке описання:

Даний варіант використання описує завдання оператором масштабу фігури.

1. Оператор обирає потрібний масштаб.
2. Програма відображає фігуру.

Варіант використання «Задати тип дискретизації»

Коротке описання:

Даний варіант використання описує вибір типу дискретизації для розбиття фігури.

1. Оператор задає тип дискретизації (1D,2D,3D)

2. Програмний комплекс дискретизує фігуру на скінчені елементи.

Варіант використання «1D- розбиття» , «2D-розбиття», «3D-розбиття»

Коротке описання:

Дані варіанти використання описують розбиття фігури для 1D,2D,3D- площини.

1. Оператор задає команду для розбиття фігур, а саме для 1D- площини.

2. Програмний комплекс виконує розбиття 1D,2D,3D- розбиття.

Варіант використання «Дискретизована фігура»

Коротке описання:

Варіант використання описує автоматичне виконання дискретизації на 1D, 2D чи на 3D скінчені елементи.

1. Програма отримує запит на розбиття фігури.

2. Програма дискретизує фігуру.

Варіант використання «Оптимізація сітки скінчених елементів»

Коротке описання:

Варіант використання описує покращення топологічної якості сітки.

1. Програма отримує запит від оператора на покращення якості сітки.

2. Програма оптимізує сітку.

Варіант використання «Збереження сітки скінчених елементів»

Коротке описання:

Варіант використання описує процес збереження побудованої моделі.

1. Програмний комплекс отримує запит на збереження моделі.

2. Програма зберігає модель по заданому шляху.



Рис.2.2. Use Case Diagram «Дискретизація фігури»

Діаграма послідовності дій (Рис.2.3.) відображає взаємодію об'єктів, впорядковану за часом. На ній показані об'єкти і класи, використовувані в сценарії, і послідовність повідомлень, якими обмінюються об'єкти, для виконання сценарію. Діаграми послідовності дій зазвичай відповідають реалізаціям прецедентів в логічному представленні системи.

Діаграма відображає послідовність повідомлень між об'єктами.



Рис2.3. Діаграма послідовності дій

Кооперативна діаграма відображає потік подій через сценарій варіанта використання. Діаграма (Рис.2.4.) загострює увагу на зв'язках між об'єктами.

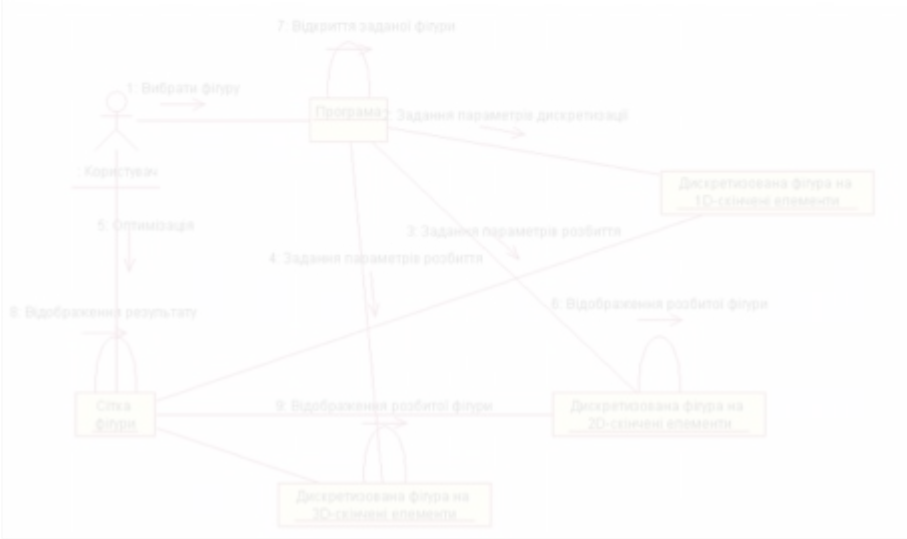


Рис.2.4.Діаграма кооперації

2.3. Фізичне представлення моделі програмної розробки

Діаграми станів визначають всі можливі стани, в яких може перебувати конкретний об'єкт, а також процес зміни станів об'єкта в результаті настання деяких подій.

На діаграмі є два спеціальних стану - початкове (start) і кінцеве (stop). Початковий стан виділено чорною точкою, він відповідає стану об'єкта, коли він тільки що був створений. Кінцевий стан позначається чорною точкою в білому кружку, він відповідає стану об'єкта перед його знищенням. На діаграмі станів може бути один і тільки один початковий стан.



Рис.2.5. Діаграма стану (Statechart diagrams)

Діаграма діяльності (Activity diagrams) деталізує особливості алгоритмічної реалізації виконання програмним комплексом операцій.



Рис.2.5. Діаграма діяльності (Activity diagrams)

2.4. Архітектура програмного забезпечення додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій

Архітектурою програмного забезпечення є структура системи, яка представляє набір компонентів, що виконують певну функцію або набір функцій. Основне призначення архітектури - організація компонентів з метою забезпечення певної функціональності.

Розглядається архітектура програмного забезпечення для автоматичної генерації розрахункових сіток для скінчено – елементного моделювання конструкцій. Розроблена схема (Рис.2.6.) з основними функціональними компонентами, які входять до складу додатку автоматичної генерації скінчено-елементного моделювання.

Архітектура програмного комплексу складається з:

- підсистеми скінчено-елементної дискретизації;
- підсистеми моделювання програмного забезпечення.

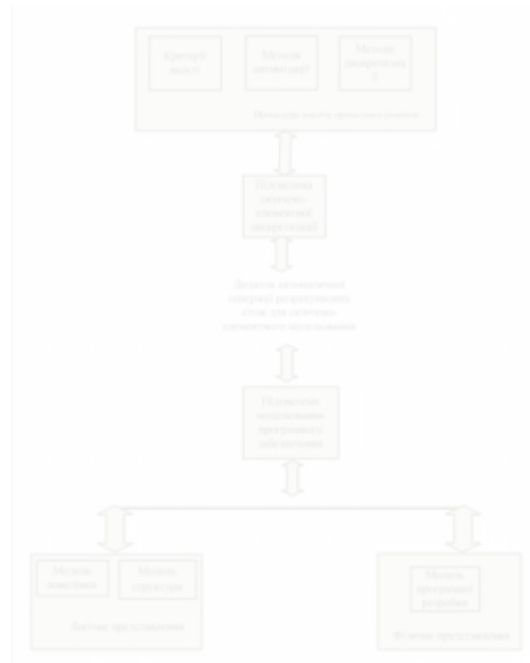


Рис.2.6. Схема компонентів програмного комплексу

2.5. Висновки до розділу 2

В цьому розділі представлено статичну модель та модель поведінки програмної розробки. А також виконано фізичне представлення моделі.

Описано архітектуру програмного забезпечення додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

РОЗДІЛ III. РОЗРОБКА ДОДАТКА ДЛЯ АВТОМАТИЗАЦІЇ ПОБУДОВИ ДИСКРЕТНОЇ (СКІНЧЕННО-ЕЛЕМЕНТНОЇ) МОДЕЛІ КОНСТРУКЦІЙ

3.1. Обґрунтування вибору середовища розробки

Microsoft Visual C++ (MSVC) — інтегроване середовище розробки додатків на мові C++, що розроблене фірмою Microsoft і поставляється як частина комплексу Microsoft Visual Studio.

Visual C++ 2010 надає потужне і гнучке середовище розробки, що дозволяє створювати додатки для Microsoft Windows і додатки, засновані на Microsoft .NET. Це середовище можна використовувати як інтегроване середовище розробки, так і в якості окремих засобів [11]. Visual C++ складається з наступних компонентів:

- Засоби компілятора Visual C++ 2010. Компілятор підтримує як традиційну розробку з використанням машинного коду, так і розробку з використанням платформ віртуальних машин, таких як середовище CLR. Visual C++ 2010 містить компілятори для цільового об'єкту x64 і Itanium. Компілятор продовжує безпосередньо підтримувати архітектуру x86 і оптимізує продуктивність коду для обох платформ.
- Бібліотеки Visual C++. Містять загальновизнану бібліотеку шаблонних класів (ATL), бібліотеки Microsoft Foundation Class (MFC) і стандартні бібліотеки, такі як стандартна бібліотека C++, яка складається з бібліотеки iostreams, бібліотеки стандартних шаблонів (STL) і бібліотеки часу виконання мови C (CRT). Бібліотека CRT включає альтернативні функції з поліпшеною безпекою для функцій з відомими проблемами безпеки. Бібліотека STL/CLR дозволяє розробникам, що використовують керований код, використовувати також і можливості бібліотеки STL. Бібліотека підтримки C++ надає нові можливості для

маршалінгу даних і спрощує написання програм, що використовують середовище CLR.

- Середовище розробки Visual C++. Середовище розробки надає всебічну підтримку при управлінні проектами та їх налаштуванні (включаючи поліпшену підтримку великих проектів), редагуванні початкового коду, перегляді початкового коду, а також потужні засоби відладки. Середовище розробки також підтримує технологію IntelliSense, яка надає при написанні коду детальні підказки, що враховують контекст.

Мова C++, що є найпопулярнішою у світі мовою рівня системи, і Visual C++ разом надають розробникові висококласний засіб світового рівня для побудови програмного забезпечення.

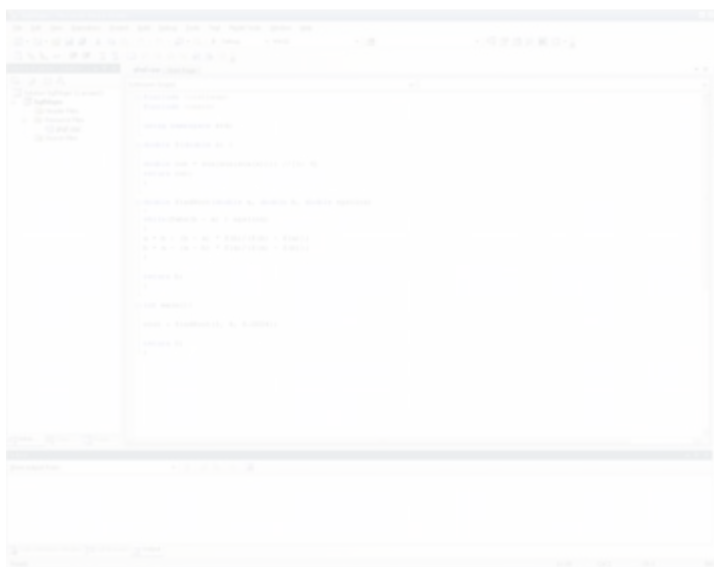


Рис. 3.1. Microsoft Visual Studio 2010

Програма реалізована на мові програмування C++ в **середовищі Visual C++ 2010**. Вибір середовища програмування обумовлений відносною простотою реалізації графічного інтерфейсу користувача (GraphicUserInterface – GUI).

При створенні засобів відображення і візуалізації використаний графічний інтерфейс OpenGL.

Бібліотека OpenGL являє собою інтерфейс програмування тривимірної графіки. Одиницею інформації є вершина, з яких створюються складніші об'єкти. Програміст створює вершини, вказує яким чином їх сполучати (лініями або багатокутниками), встановлює координати і параметри камери та ламп, а бібліотека OpenGL бере на себе роботу по створенню зображення на екрані. OpenGL ідеально підходить для програмістів, яким необхідно створити невелику тривимірну сцену і не замислюватися про деталі реалізації алгоритмів тривимірної графіки.

З погляду архітектури, графічна система OpenGL є конвеєром, що складається з декількох етапів обробки даних:

- Апроксимація кривих і поверхонь;
- Обробка вершин і збірка примітивів;
- Растеризація і обробка фрагментів;
- Операції над пікселями;
- Підготовка текстури;
- Передача даних в буфер кадру.

3.2. Розробка інтерфейсу

Дуже важливу роль в ефективності роботи програми грає правильно розроблений інтерфейс. Саме від цього буде залежати виконання декількох з основних вимог - зручність і простота в освоєнні. Складний, перевантажений інтерфейс може зменшити ефективність роботи з препроцесором. Головне правило, від якого варто відштовхуватися - інтерфейс не повинен заважати роботі користувача й не повинен відволікати його увагу від роботи, одночасно не втрачаючи при цьому функціональності.

Розроблений програмний комплекс виконує дискретизацію на скінченні елементи досліджуваної області в конструкціях для елементів чисельного розрахунку напружено-деформованого стану деформівного тіла.

Має змогу:

38

- відкривати файли: Gmsh geometry .geo - файл геометрії GMSH; STEP і IGES - відомі формати файлів геометрії, підтримувані багатьма CAD –пакетами;
- автоматично виконувати дискретизацію на 1D, 2D чи на 3D скінчені елементи;
- візуалізує дискретизацію на 1D, 2D чи на 3D скінчені елементи;
- візуалізує нумерацію скінчених елементів;
- автоматизує рекомбінацію сітки скінчених елементів;
- виконувати оптимізацію сітки скінчених елементів.

Програмний комплекс зберігає в файл геометрію з дискретизацією 1D, 2D чи 3D скінчені елементи.

Програмний комплекс має досить зручний інтерфейс. В ньому передбачена можливість виконувати дискретизацією на 1D, 2D чи 3D скінчені елементи для розбиття криволінійних, просторових поверхонь і поверхонь, що обмежують об'ємні тіла та об'ємні тіла;

Програма має змогу виконувати оптимізацію скінчених елементів та забезпечує збереження в файл сітку скінчених елементів конструкцій різних форм.

Інтерфейс програми представлений в вигляді поєднання двох вікон (Рис.3.3.). Одне з них являється Графічним Вікном, де створюється модель і редагується. А інше містить в собі основне Меню Програми та набір команд для роботи з фігурами.



Рис.3.2. Інтерфейс програми

В нижній частині Графічного Вікна знаходиться Рядок Стану (Рис.3.3.).

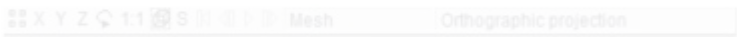


Рис.3.3. Рядок Стану

Він містить команди для керування зображенням і інформацію про дію активної команди. (Табл.3.1.)

Таблиця 3.1.-Команди Рядка Стану

Зображення піктограми	Комбінація клавіш	Дія
	Alt+x, Alt+Shift+x	Вид моделі відносно осі X.
	Alt+y, Alt+Shift+y	Вид моделі відносно осі Y.
	Alt+z, Alt+Shift+z	Вид моделі відносно осі Z.
	Shift Alt	Поворот моделі на 90 за часовою стрілкою та проти часової. Синхронізація повороту.
	Alt	Встановлення масштабу. Синхронізація масштабу.
	Alt+o, Alt+Shift+o	Перемикання режиму проекції.
	Esc	Перемикач миші Вкл/Викл.

Вікно Меню Програми містить в собі основне меню програми та набір команд для роботи з фігурами. У верхній ділянці розташований рядок падаючих меню. В нього входять File, Tools.

За допомогою меню Files можна :

- відкрити потрібний файл ;
- об'єднати декілька фігур;
- видалити непотрібні елементи;
- відкрити одразу декілька вікон для роботи чи розділити активоване;
- зберегти побудовану модель;
- зберегти її безпосередньо отриману сітку;
- зберегти опції;
- вийти з програми.

За допомогою меню Tools можна :

- задати потрібні параметри моделі;
- зробити видимою будь-яку частину фігури для зручності роботи;
- задати координати обертання.

1D, 2D, 3D-розбиття виконується за допомогою команд з Головного меню.

Після відкриття фігури натискаємо на кнопку "2D". Результат показаний на рис. 3.1. Вийшло те, що називають "триангуляція", причому обертає на себе увагу місцеве згущування сітки в районі точки №3 внаслідок прийняття для неї відповідного значення характеристичної довжини.

Рис. 3.4.Розбиття прямокутника на СЕ

Отже, у нас вийшли трьохвузлові елементи першого порядку. А якщо ми натиснемо на кнопку "Second Order" на екрані Графічного Вікна будуть вже шести-вузлові елементи (трикутні, з проміжними вузлами), як відомо - на порядок точніші з розрахункової точки зору, а отже що не вимагають великого подрібнення для отримання заданої точності. На практиці сітка згущується в зонах концентрації напруги, там де напруга змінюється різко на невеликому відрізьку. Там, де градієнт напруги не високий, можна обійтися грубішою сіткою, а дрібнити усе однаково - непродуктивно з точки зору точності і економії машинних ресурсів.

Якщо ми зайдемо на вкладку: Mesh => Define => Recombine, виділимо поверхню, після закінчення виділення натиснемо "e" і розіб'ємо заново, то отримаємо результат, показаний на рис. 3.2, а в список команд додається рядок: Recombine Surface {6}

Рис. 3.5. Рекомбінована сітка

Отже, ми скористалися для розбиття на СЕ кнопкою "2D". "2D" - не обов'язково означає плоску сітку, нею треба користуватися і для розбиття криволінійних, просторових поверхонь і поверхонь, що обмежують об'ємні тіла.

Для розбиття об'ємної фігури потрібно її вибрати за допомогою меню File. (Рис.3.5.)

Рис.3.6. Вибір фігури для розбиття

За допомогою кнопки 2D фігура розбивається на скінченні елементи в 2D-площині (Рис.3.6.).

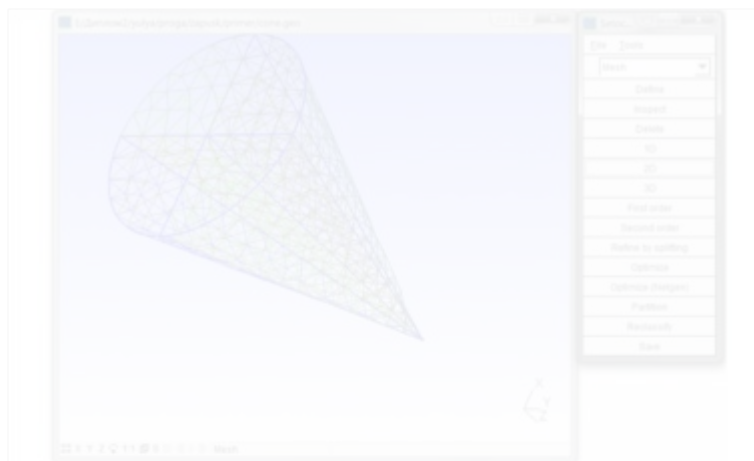


Рис.3.6. 2D-розбиття

При натисканні кнопки 3D виконується розбиття на елементи в 3D-площині (Рис.3.7.).

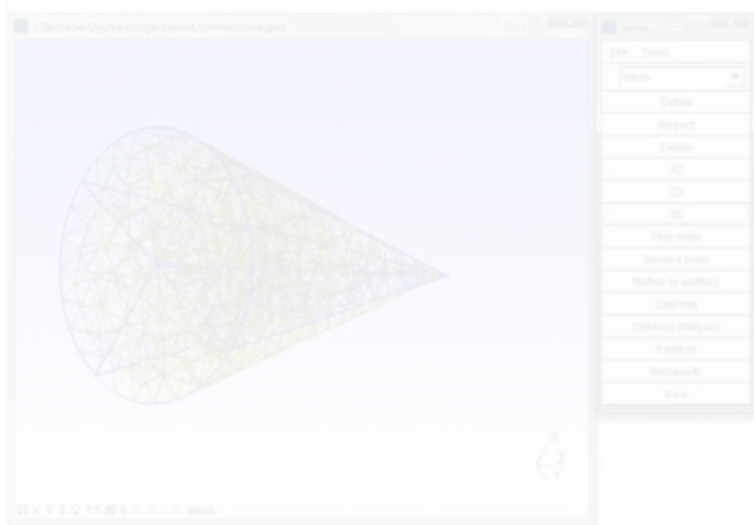


Рис.3.7. 3D-розбиття

Розглянемо опції програмного додатку які принципово впливають на побудову сітки, якість її елементів, а також їх збереження. У дужках після назви опції стоїть значення за замовчуванням.

Geometry.ExtrudeReturnLateralEntities (=1) - чи потрібно додавати додаткові номери в масив, що повертається командою Extrude.

Geometry.ExtrudeSplinePoints (=5) - кількість контрольних точок для сплайнів, створених внаслідок витягування.

Geometry.NumSubEdges (=20) - кількість відрізків між контрольними точками при відображенні кривих (ця опція не стосується побудови сітки, проте дозволяє сильно поліпшити відображення складних кривих).

Geometry.OCCFixSmallEdges (=1) - фіксувати малі ребра в IGES, STEP і BRep моделях (важлива опція при експорті моделей з інших CAD програм).

Geometry.OCCFixSmallFaces (=1) - фіксувати малі грані в IGES, STEP і BRep моделях (важлива опція при експорті моделей з інших CAD програм).

Geometry.Tolerance (=1e-06) - геометрична точність (допустиме відхилення).

Mesh.Algorithm3D (=1 - алгоритм побудови тетраедральної сітки (1-Delaunay, 2-Frontal). Алгоритм Delaunay найбільш стійкий і швидкий, однак іноді він змінює поверхневу сітку (спочатку будує одновимірну сітку, потім засновану на ній двовимірну, а потім засновану на двовимірній тривимірну сітку) і не підходить для створення змішаної структурованої/неструктурованою сітки. Для цих випадків підходить Frontal алгоритм. Якість елементів, вироблених цими алгоритмами, приблизно однакове.

Mesh.Binary (=0) - записувати сітку в бінарному форматі.

Mesh.CharacteristicLengthFactor (=1) - множник характеристичної довжини (застосовується, щоб подрібнити/згрубіть всі елементи сітки в рівній мірі).

Mesh.CharacteristicLengthMin (=0) - мінімальний розмір елемента сітки.

45

Mesh.CharacteristicLengthMax (=1e+22) - максимальний розмір елемента сітки.

Mesh.CpuTime (=0) - час (у секундах), відведений для побудови сітки.

Mesh.ElementOrder (=1) - порядок елементів сітки (1 - лінійні елементи; доступні також 2, 3, 4 і 5 порядки).

Mesh.MinimumCirclePoints (=7) - мінімальна кількість точок для апроксимації окружності.

Mesh.MinimumCurvePoints (=3) - мінімальна кількість точок для апроксимації кривої лінії.

Mesh.NumSubEdges (=2) - кількість ребер розбиття при відображенні елементів високих порядків. Ця опція важлива тільки з точки зору візуалізації. Проілюструємо це на прикладі розбиття кола грубою сіткою другого порядку (зліва - кількість ребер = 2, праворуч - кількість ребер = 5):

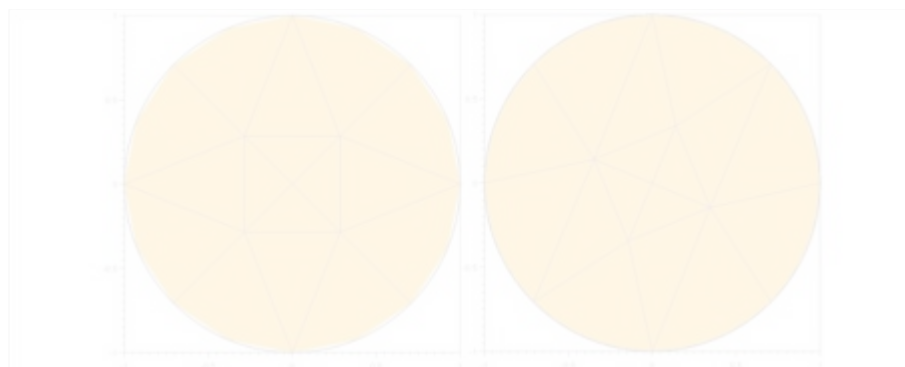


Рис. 3.8. Розбиття кола грубою сіткою

Mesh.SaveAll (=0) - зберігати всі елементи сітки, незалежно від фізичних об'єктів.

Mesh.SecondOrderIncomplete (=0) - створювати елементи неповного другого порядку (8 - вузлові чотирикутники, 20 - вузлові шестигранники і т.д.).

Mesh.SecondOrderLinear (=0) - чи повинні вершини елементів другого порядку бути отримані лінійною інтерполяцією (у цьому випадку, додаткові вузли ставляться строго на середину ребра елемента. Інакше, ребра

ламаються на два, а додаткові вузли зміщуються для більш точної апроксимації). Проілюструємо це (ліворуч опція відключена, праворуч - включена):



Рис. 3.9. Лінійна інтерполяція

За допомогою команди Optimize відбувається оптимізація сітки, тобто поліпшення топологічної якості сітки (Рис.3.10).



Рис.3.10. Оптимізація сітки

Зупинимось на опціях, які принципово впливають на якість сітки, її елементів, а також їх збереження. У дужках після назви опції стоїть значення за замовчуванням.

Mesh.Optimize (=0) - оптимізувати сітку для підвищення якості тетраедральних елементів.

Mesh.OptimizeNetgen (=0) - оптимізувати сітку для підвищення якості тетраедральних елементів, використовуючи алгоритм **Netgen'a**.

Mesh.RecombinationAlgorithm (= 0) - алгоритм рекомбінування (0 - стандартний, 1 - blossom).

Mesh.Smoothing (=1) - кількість кроків згладжування, що застосовується до підсумкової сітки.

Програмний додаток нічого сам не обчислює, тому він має бути пов'язаний з іншими програмами за допомогою створюваних і прочитуваних ним файлів.

Зробити це можна через випадне меню "File" Вікна Меню, вибравши "Open" або "Save As" (рис. 3.11).



Рис. 3.11. Випадне меню "File": "Open" та "Save As"

Розглянемо деякі формати файлів по їх розширеннях:

- Gmsh geometry .geo - файл геометрії GMSH.
- STEP і IGES - відомі формати файлів геометрія, підтримувана багатьма CAD -пакетами. Імпорт STEP і IGES, розроблений на основі відкритої бібліотеки OpenCascade дозволяє завантажувати в програму моделі, побудовані в сторонніх CAD -ах (зворотне вивантаження моделі з .geo в ці формати доки не передбачена), в т.ч. у відкритому пакеті Salome.
- STL surface mesh - поширений формат для передачі поверхонь через трикутні сітки.
- .msh файл - файл сітки генератора.

Msh-файл складається з секції з інформацією про формат файлу (MeshFormat) і секцій. Дана інформація описує вузли (Nodes) й елементи (Elements) сітки:

```
$MeshFormat
$EndMeshFormat
$Nodes
nNodes
iNode x y z
$EndNodes
$Elements
nElements
iElement type nTags tags nodes
$EndElements
```

В даному прикладі:

nNodes – кількість вузлів сітки,

i – індекс вузла,

x, y, z – координати вузла,

nElements – кількість елементів,

type – геометричний тип елемента (таблиця 3.2),

nTags – кількість тегів,

tags – список тегів,

nodes – список вузлів з яких складається елемент.

За замовчуванням, у msh-файлі задається три теги: перший вказує номер фізичної сутності, до якої даний елемент належить, другий геометричну область, третє - номер частини сітки, до якої належить елемент.

Таблиця 3.2. – Геометричні типи елементів

Номер	Геометричний тип
1	Лінія
2	Трикутник
3	Чотирикутник
4	Тетраедр
5	Гексаедр

3.3. Висновки до розділу 3

Програма реалізована на мові програмування C++ в середовищі Visual C++ 2010, при створенні засобів відображення і візуалізації. Описано алгоритм генерації сіток. У додатку доступно створення 1- 2 - і 3-х мірних сіток, і їх оптимізація. Програма реалізується на мові програмування C++ в середовищі Visual C++ 2010. При створенні засобів відображення та візуалізації використовується графічний інтерфейс OpenGL.

ВИСНОВКИ

В магістерській роботі було розглянуто розробку додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

Для досягнення поставленої мети були вирішені наступні задачі:

- Проаналізовано основні поширені методи й алгоритми дискретизації плоских та просторових областей;
- Проведено моделювання та аналіз програмного забезпечення додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій, розроблено UML- моделі логічного і фізичного уявлень;
- Розглянуто програмну реалізацію додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

В першому розділі проаналізовано основні підходи, що застосовуються в сучасних САПР для твердотільного моделювання геометричних об'єктів, а також основні поширені методи й алгоритми дискретизації плоских та просторових областей. Найпоширеніші алгоритми дискретизації можна розбити на 2 частини: побудову первинної дискретизації та її оптимізації. У додатку автоматичної генерації розрахункових сіток для скінченно-елементного моделювання для первинної дискретизації області прийнято застосовувати модифікований алгоритм Ватсона-Лавсона. Для оптимізації отриманої скінченно-елементної сітки - алгоритм Рапперта в плоскому випадку та Шевчука в тривимірному.

50

В другому розділі промодельоване та проаналізовано програмне забезпечення для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій. Розроблено UML- діаграми представлення статичної моделі структури, представлення моделі поведінки та фізичне представлення моделей програмної розробки.

У третьому розділі аргументується вибір середовища розробки додатку і мови програмування. Описано алгоритм генерації сіток. У додатку доступно створення 1- 2 - і 3-х мірних сіток, і їх оптимізація. Програма реалізується на мові програмування C ++ в середовищі Visual C ++ 2010. При створенні засобів відображення та візуалізації використовується графічний інтерфейс OpenGL.

Matches

Internet sources

3

1 <https://www.bibliofond.ru/view.aspx?id=787505>

1.5%

3 <https://www.sesiya.ru/otchet-po-praktike/programmirovaniye/proektuvannya-ta-rozrobka-programnogo-zabezpechennya-gri-sh..>

0.29%

4 http://4ua.co.ua/economy/za2bc68b5c53a88421316c26_0.html

0.11%

Library sources

1

2 **Student submission** File ID: **1013147472** Institution: **Luhansk Taras Shevchenko National University**

0.79%

Quotes

Quotes

2

- 1 Use Case Diagram «Дискретизація фігури» Діаграма послідовності дій (Рис.2.3.) відображає взаємодію об'єктів, впорядковану за часом.
- 2 3.9. Лінійна інтерполяція За допомогою команди Optimize відбувається оптимізація сітки, тобто поліпшення топологічної якості сітки (Рис.3.10).